NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

RECOGNITION OF SHIP TYPES FROM AN INFRARED IMAGE USING MOMENT INVARIANTS AND NEURAL NETWORKS

by

Jorge Amaral Alves

March 2001

Thesis Advisor: Neil C. Rowe
Second Reader: Robert B. McGhee

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE March 2001	3. REPORT TYPE AND DATES COVERED Master's Thesis	

4. TITLE AND SUBTITLE
Recognition of Ship Types From an Infrared Image Using Moment Invariants and Neural Networks

5. FUNDING NUMBERS

6. AUTHOR(S)
Alves, Jorge Amaral

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Brazilian Naval Commission
5130 MacArthur Blvd. N. W.
Washington, D.C. 20016-3344

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense, the U.S. Government or the Brazilian Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Autonomous object recognition is an active area of interest for military and commercial applications: Given an input image from an infrared or range sensor, find interesting objects in those images and then classify those objects. In this work, automatic target recognition of ship types in an infrared image is explored. The first phase segments the original infrared image in order to obtain the ship silhouette. The second phase calculates moment functions of those silhouettes that guarantee invariance with respect to translation, rotation and scale. The third phase applies those invariant features to a backpropagation neural network and classifies the ship as one of five types. The algorithm was implemented and experimentally validated using both simulated three-dimensional ship model images and real images derived from video of an AN/AAS-44V Forward Looking Infrared (FLIR) sensor.

14. SUBJECT TERMS
Automatic target recognition, artificial neural network, infrared image recognition, moment invariants

15. NUMBER OF PAGES
120

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

RECOGNITION OF SHIP TYPES FROM AN INFRARED IMAGE USING MOMENT INVARIANTS AND NEURAL NETWORKS

Jorge Amaral Alves
Lieutenant Commander, Brazilian Navy
B.S., Brazilian Naval Academy, 1987
B.S.E.E., University of Sao Paulo, 1993
M.S.E.E., Federal University of Rio de Janeiro, 1998

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
March 2001**

Author: _____

Jorge Amaral Alves

Approved by: _____

Neil C. Rowe, Thesis Advisor

_____

Robert B. McGhee, Second Reader

_____

Dan Boger, Chairman
Department of Computer Science

ABSTRACT

Autonomous object recognition is an active area of interest for military and commercial applications: Given an input image from an infrared or range sensor, find interesting objects in those images and then classify those objects. In this work, automatic target recognition of ship types in an infrared image is explored. The first phase segments the original infrared image in order to obtain the ship silhouette. The second phase calculates moment functions of those silhouettes that guarantee invariance with respect to translation, rotation and scale. The third phase applies those invariant features to a backpropagation neural network and classifies the ship as one of the five types. The algorithm was implemented and experimentally validated using both simulated three-dimensional ship model images and real images derived from video of an AN/AAS-44V Forward Looking Infrared (FLIR) sensor.

DISCLAIMER

The algorithms and computer programs developed in this research were not exercised for all possible cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

ACKNOWLEDGMENT

I wish to express my gratitude to those who contributed to the successful and timely completion of this research. First I would like to thank Professor Robert McGhee for his help on indicating good algorithms for image feature extraction; also for his dedication and thoroughness in reviewing the text and the technical content of this work. Special recognition is due Mrs. Jean Brennan, the Computer Science Department assistant, for his outstanding administrative work.

My deepest gratitude goes to Professor Neil Rowe for his patience, constant guidance, cooperation, and most of all, for being a friend whose experience I could always count upon.

Finally, I would like to thank my wife Margareth, my son Edgard and my daughter Caroline for their support and understanding as I work in this thesis.

I.   INTRODUCTION

A.   AUTOMATIC TARGET RECOGNITION

Automatic target recognition (ATR) is a technological discipline that deals with the understanding, design, development, and production of techniques and hardware for the classification of objects of interest as they are sensed by remote means, either actively or passively. During the past several decades, numerous attempts have been made to create such systems [Ref. 1-6]. However, progress in ATR has been slow [Ref. 7] because some new problems have appeared. For example, the vision problem pushes the fields of artificial intelligence, neural networks, microelectronics, sensors, and computer science to their limits.

In any pattern recognition application, it is important to select features that adequately and uniquely describe the objects to be recognized. Moreover, the features associated with an object should be invariant with respect to the position, rotation, and scale of that object in the field of view. Thus the ideal recognition system is robust to orientation variations, scale variations and boundary perturbations [Ref. 8].

Our proposed approach is to use the moment invariants [Ref. 9] for the set of features to quantify the object. The thesis reports the mathematical foundation of two-dimensional moment invariants and shows that recognition schemes based on them could be truly position, size and orientation-independent. Since the moments are global features, application of such a feature space is limited to images with minimal background and scenes containing only one object. Ships on the open sea are appropriate for such feature spaces [Ref. 10]. The moment invariants are used to construct a feature vector of low dimension, and recognition is performed using this feature vector applied to a trained artificial neural network classifier.

B.   APPLICATIONS OF AUTOMATIC OBJECT RECOGNITION

Automatic object recognition has diverse applications in numerous fields of science and technology and is permeating many aspects of military and civilian

industries. It is popular within the field of robotic vision because of the limited domains and the controllability of the environment in which they are used [Ref. 7].

In military applications and specifically naval applications, electro-optic and infrared sensors have been connected to weapon systems. In several cases an ATR algorithm is responsible for discriminating a target from a non-target object, enabling the possible target destruction. Other systems address classification tasks where targets types are determined. Another example is a Forward Looking Infrared (FLIR) sensor combined with image-processing hardware for discriminating tanks from trucks, bushes, and other environmental objects.

C.   PROJECT GOALS

In this thesis, a fast and robust system is presented that classifies ships seen from an arbitrary viewpoint and range in three-dimensional space. The approach is concerned with segmented rigid bodies viewed without occlusion from other objects. However, self-occlusion due to change of viewpoint is allowed.

Although object separation from background is a challenging task in general, our application can be carried out with relative ease. This is the case because a ship has usually a clear contrast with the background in Forward Looking Infrared (FLIR) imagery. This greatly simplifies ship classification.

Our approach is model-based, meaning that the kinds of objects to be recognized are known in advance and can be summarized in a set of models. The specific model database we have implemented contains five classes of ships: destroyer, frigate, aircraft carrier, research ship and merchant ship. This database was used in the training phase. For each ship model and viewpoint, a silhouette was extracted and a moment-invariant signature calculated consisting of a twelve-element feature vector.

Then for a ship image of unknown type, we compute its signature. Classification is done using an artificial neural network. The neural-net classifier's generalization capabilities are used to group the moment-invariant signatures, corresponding to different views of an object, into a single ship type class.

2

The proposed scheme is summarized in Figure 1 below.
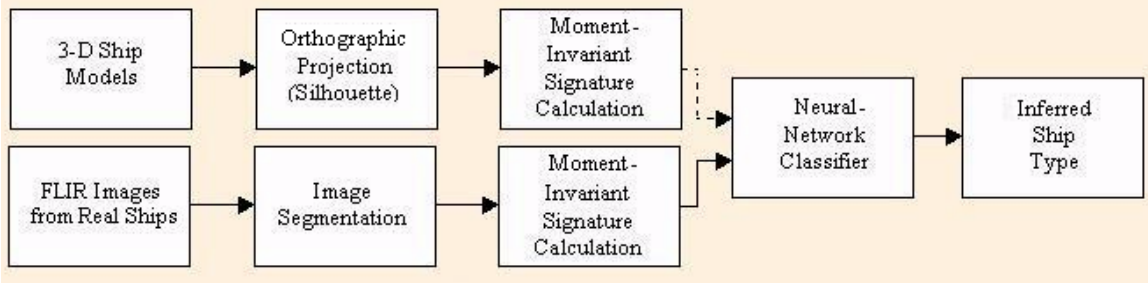
Figure 1: Processing scheme of moment invariant recognition.

This thesis has been organized as follows: in Chapter II we present techniques for feature extraction followed by mathematical foundations of moment invariants. In Chapter III we present an artificial neural network as a classifier and the backpropagation learning rule. Chapter IV details the input image database and the training and testing of our system. Chapter V summarizes the results from the experimentation using simulated images from three-dimensional ship models and real ship images from FLIR sensors. Chapter VI contains concluding remarks.

THIS PAGE INTENTIONALLY LEFT BLANK

II. FEATURE SELECTION AND MOMENT INVARIANTS

A. OVERVIEW

An old adage says "Good Features make Good Recognizers" [Ref. 11]. This is true whether your recognizer is using an artificial network or a statistical based decision mechanism. So our project paid careful attention to feature extraction from ship images.

Many approaches have been advocated for features in automatic target recognition. Present methods can be categorized as either global or local. Global methods use global features of an object boundary or of an equivalent representation. Such techniques are the Fourier descriptors (FD) [Refs. 1, 5], moments [Ref. 12] and autoregressive models [Ref. 13]. Local methods use features such as critical points [Ref. 14] or high-resolution pursuit (HRP) [Refs. 8, 15].

For global-based approaches, there is a wide variety of published literature similar to the approach described herein. Global methods have the disadvantage that a small distortion in a section of a boundary of an object will result in changes to all global features.

One early work is Dudani et al [Ref. 16], which used moment invariants for feature extraction and a probabilistic approach for the classification of airplanes. Dudani used six different aircraft types and the images were based on physical models. His training set was based on over 3000 images taken in a 140° by 90° sector. The testing set contained 132 images (22 images of each of the six classes) obtained at random viewing aspects. The classification accuracy achieved in this six-class problem was 95%.

Later, Wallace and Wintz [Ref. 17] propose a technique similar to Dudani's with Fourier Descriptor (FD) of the silhouette boundary as features. The Fourier descriptor is one method of describing the shape of a closed figure. Wallace and Wintz used a graphics program to test their algorithm implementing three-dimensional models for six different aircrafts. The graphics program approximated each airplane by using 50-100 planes. Again, the evaluation was done using a randomly selected set and comparing to the library of projections. However, they used only 143 projections for training (9.9 times less than that used by Dudani et al) and the aircraft outlines were taken from a sector of

180° by 180°. Wallace and Wintz considered a bigger sector trying to avoid Dudani's approach, because if we delete the angles near the front view and rear view of the aircraft the problem is much easier: shapes vary much more with slight rotations when viewed almost edgewise [Ref. 17]. The maximum classification accuracy achieved by Wallace and Wintz was 88.0%.

Reeves et al [Ref. 18] presents a geometrical-moment approach using moments of the image that are normalized with respect to scale, translation and rotation. They call them "standard moments". The experiments described there were based on the same software used by Wallace and Wintz. They also used the same six types of airplanes, the same training set and the same testing set. However, they have chosen the moment feature representation because Fourier descriptors (FD) are particularly sensitive to perturbations in the object boundary. For example, the FD's for the image of a disk differ greatly from those for a disk with a tiny wedge missing. Reeves et al used two classification criteria: the minimum Euclidean distance and the minimum Euclidean distance after "variance balancing". The best classification result was 93%.

More recent work of Khotanzad [Ref. 19] used global features derived from complex orthogonal Pseudo-Zernike Moments (PZM). Khotanzad tested the performance of PZM by recognizing 26 uppercase English characters (A to Z), typed and handwritten. The database contained 624 images corresponding to 24 images per character. These images were generated with arbitrarily varying scales, orientations, and translations. The available samples were divided into halves. The first half was used for training and the second for testing. There were 12 training images and 12 testing images per character. His neural network classifier formed by 45 input nodes, 26 output nodes and 40 hidden nodes got 100% of classification accuracy.

Systems using local features perform well in the presence of noise, distortion or partial occlusion. The effects on an isolated region of the contour alter only the local features associated with that region, leaving all the other local features unaffected. However, the choice of representative local features is not trivial and the recognition process based on local features is more computationally intensive and time consuming [Ref. 2].

B. MOMENT INVARIANTS

Moment invariants are a reliable and versatile way to construct a feature vector of low dimension as the basis for the neural-network classifier. Moments have been used as pattern features in a number of applications [Ref. 9, 20] to recognize two-dimensional image patterns.

$$p, q = 0,1,2,... \qquad (1)$$

The regular moments m_{pq} of a digital image pattern represented by f(x,y) are defined as:

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y) \qquad (2)$$

Hu [Ref. 9] first introduced moments as image-recognition features. Using nonlinear combinations of normalized central moments, he derived seven invariant moments, which have the desirable property of being invariant under image translation, scaling and rotation. The classic central moments that have the property of translation invariance are:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q \qquad p, q = 0,1,2,...$$
$$where \; \bar{x} = \frac{m_{10}}{m_{00}} \; and \; \bar{y} = \frac{m_{01}}{m_{00}}$$

Hu discovered these moments M1, M2, …, M7, are invariant under translation and rotation:

$$M1 = \mu_{20} + \mu_{02} \qquad (3)$$

$$M2 = (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2 \qquad (4)$$

$$M3 = (\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 \qquad (5)$$

$$M4 = (\mu_{30} + \mu_{12})^2 + (3\mu_{21} + \mu_{03})^2 \qquad (6)$$

$$M5 = (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2]$$
$$+ (3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] \qquad (7)$$

$$M6 = (\mu_{20} - \mu_{02})[\ (\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2\ ] + 4\mu_{11}(\mu_{30} + \mu_{12})(\ \mu_{21} + \mu_{03}) \qquad (8)$$

$$M7 = (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[\ (\mu_{30} + \mu_{12})^2 - 3(\ \mu_{21} + \mu_{03})^2\ ]$$

$$- (\mu_{30} - 3\mu_{12})\ (\mu_{21} - \mu_{03})[\ 3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2\ ] \qquad (9)$$

The functions M1 through M6 are invariant under rotation, reflection, or a combination of rotation and reflection. This property helps to simplify the range of all distinct views of the ships as explained in section D of chapter IV.

The above moments can be normalized to become invariant under a scale change by using the radius of gyration r of a planar pattern [Ref. 16]:

$$r = (\mu_{20} + \mu_{02})^{1/2} \qquad (10)$$

The radius of gyration for a particular object from a particular angle of view is directly proportional to the size of the image or inversely proportional to the distance B of the object along the optical axis:

$$(\mu_{20} + \mu_{02})^{1/2}\ B = \text{constant} \qquad (11)$$

Therefore, the radius of gyration r can normalize the moment functions M2 through M7 to obtain size invariance, what Hu called the "normalized central moments":

$$M1' = (\mu_{20} + \mu_{02})^{1/2}\ B = r\ B \qquad (12)$$

$$M2' = M2\ /\ r^4 \qquad (13)$$

$$M3' = M3\ /\ r^6 \qquad (14)$$

$$M4' = M4\ /\ r^6 \qquad (15)$$

$$M5' = M5\ /\ r^{12} \qquad (16)$$

$$M6' = M6\ /\ r^8 \qquad (17)$$

$$M7' = M7\ /\ r^{12} \qquad (18)$$

C.   FEATURE VECTOR

The above moments of an object can be computed for both the image boundary and the solid silhouette. Minute details such as the shape of the stacks of a ship are better characterized by the moments from the boundary. Gross structural features of the ship are better characterized by moments derived of silhouette; also, these moments are less susceptible to noise [Ref. 16].

In our system, two sets of six moment invariant functions (M2', M3', M4', M5', M6' and M7'), six from the boundary and six from the silhouette, were computed. As the distance B of the object along the optical axis was not known, the M1' component was not used. The twelve-component feature vector was sent to the neural network classifier for the recognition phase.

THIS PAGE INTENTIONALLY LEFT BLANK

III. THE ARTIFICIAL NEURAL NETWORK (ANN) CLASSIFIER

A. MOTIVATION

Over the past few years, an explosion of interest in ANN models and their applications has occurred [Ref. 21, 22, 23]. ANNs posses a number of properties which make them particularly suited to complex classification problems [Ref. 22, 25, 26]. Unlike traditional classifiers, ANN models can examine numerous competing hypotheses simultaneously using massive interconnections among many simple processing elements. In addition, ANNs perform extremely well under noise and distortion.

The implementation of a model-based target recognition scheme using ANNs seems to be attractive. First of all, ANNs provide their own way to represent the knowledge that they store [Ref. 27]. In addition, the complexity and the computational burden increase slowly as the number of data models increases.

Although ANN's performance is excellent, many researchers still criticize ANNs because they can require much training time before they can perform a specific task. However, in our automatic target recognition classifier, the recognition phase is of far more importance and it must run as quickly and accurately as possible; the training phase can be performed off-line.

In this thesis specifically, a three-layer perceptron neural network [Ref. 28] trained with the backpropagation learning rule [Ref. 29] was implemented. In this scheme, expensive storage of a multiview database is not needed since during training the neural net extracts all the relevant information from the library. Also, due to the generalization capability of the neural net, good results can be obtained even with a small number of views in the library.

B. MULTILAYER ANN AND THE BACKPROPAGATION RULE

Artificial neural networks were developed by modeling a biological neuron. A generic neuron is formed by "cell body", "dendrites" and "axon". The electrical signals arrive in a neuron by the dendrites and are passed to the cell body where they are added. If a threshold is achieved, the neuron is activated and the information is passed to the axon. The axon is the transmission line of the neuron. The axon will pass the information to the chemical synapses connections. The learning process will be responsible for increasing the synaptic strength, which measures the degree of coupling between two neurons.

Many neuron models appear in the literature. The beginning of the development of neural network models is related to the paper of Warren McCulloch and Walter Pitts published in 1943. They studied the implementation of logical functions using artificial neurons. The mathematical model of the neuron proposed by McCulloch-Pitts, shown in Figure 2 below, assumes the function realized by the cell body as being a "step function" applied to the summation of the weighted inputs. The weights control the importance of each input.
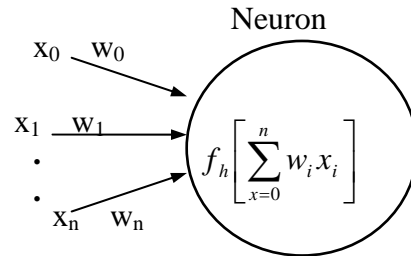
Figure 2: McCulloch-Pitts Model.

The McCulloch-Pitts model applied to a single layer of neurons (perceptrons) cannot solve problems where the inputs cannot be linearly separated. The PDP group in their collection of papers [Ref. 28] proposed modifications to the previous model. The step function was replaced by a function that is monotonic, differentiable and smooth (often implemented by a sigmoid). The learning algorithm used is "backpropagation".

Artificial neural networks (ANN) are specified by the topology of the network, the characteristics of the nodes (neurons) and the learning algorithm. The topology of a multilayer ANN is a structured hierarchical layered network as shown in Figure 3 below:
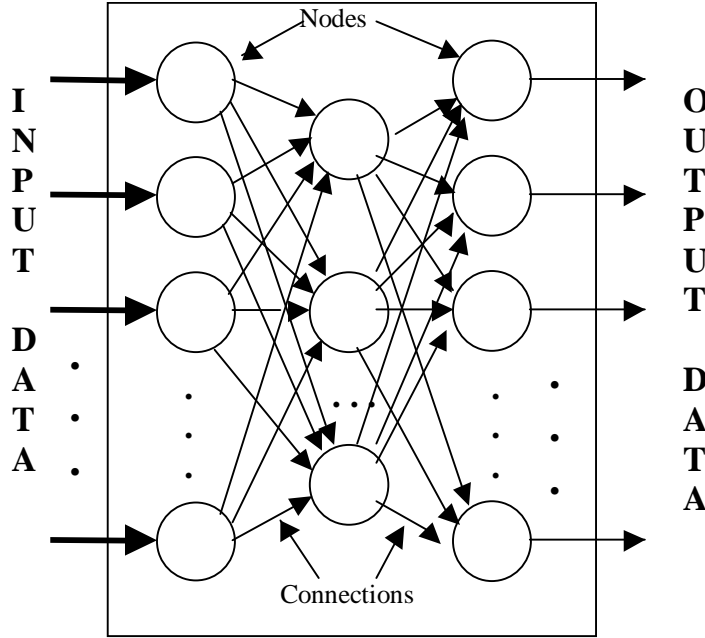
Figure 3: Multilayer neural-network graph.

It consists of several layers of nodes, and usually an input layer and an output layer. Between the input layer and the output layer, we have one or more "hidden" layers of nodes. Hidden nodes often represent domain knowledge useful for solving recognition tasks [Ref. 27]. Generally, each node in one layer is interconnected with all the nodes in adjacent layers with connections (synapses). Each connection is associated with a weight, which measures the degree of interaction between the corresponding nodes.

A general L-layered feed-forward artificial neural network consists of N_0 input nodes and N_L output nodes. The number of nodes in the hidden layers is N_k for $1<k<L-1$. In this notation, the input layer is not counted as a layer. So an L-layer feed-forward artificial neural network has L-1 hidden layers and the Lth layer is the output layer. In this thesis, we implemented a 2-layered (L=2) feed-forward artificial neural network with $N_0=12$ (the moment invariant feature vector) and $N_2=5$ (five ship types). The number of

hidden nodes was found in order to maximize the neural net performance, as detailed in Chapter V.

The algorithms for multilayer ANN processing can be divided into two phases: retrieving and learning. In the retrieving phase of the algorithm, information flows from the input layer through the hidden layers to the output layer. The nodes update their own activation values based on the system dynamics. In the learning phase, modification of the weights corresponding to the connection edges takes place. In this thesis, the popular backpropagation rule [Ref. 28] learning algorithm is used. This algorithm performs supervised learning; in each step it adjusts the connection weights, minimizing the mean-square error between the target value (the desired) and the output value (the actual) if the network.

During the retrieving phase, we present continuous valued input data x_1, x_2, …, x_{n0} called exemplar patterns and the corresponding desired output data t_1, t_2, …, t_{nL} called target patterns. Input data are propagated forward through the network, which computes the activation value for each node, until the output layer is reached.

The learning phase involves a backward pass through the network during which the error signals produced at the output layer are passed to each node in the network and appropriate weight changes are made. For each weight, the gradient of the output error with respect to that weight is computed. The weight is changed in the direction that reduces the error.

IV. EXPERIMENT DESCRIPTION

A. PROGRAMMING ENVIRONMENT

Our Automatic Recognition Algorithm was based on programs written in MATLAB 5.3.0 from MathWorks. MATLAB is a complete computing environment for the interactive analysis and visualization of data, integrating an array-oriented language with mathematical analysis and graphical display techniques. The neural-network programs used in this thesis were implemented using functions from the MATLAB Neural Network Toolbox. The three-dimensional model and all image analysis were performed using the MATLAB Image Processing Toolbox.

Although MATLAB is an interpretative language, it is possible to translate all MATLAB source codes into C code and create executable files using the MATLAB C Compiler. Consequently, our Target Recognition System could be used in a real-time application.

B. THE THREE-DIMENSIONAL SHIP MODEL DATABASE

1. The Three-Dimensional Ship Modeling

This section will describe our implementation of three-dimensional ship models. The three-dimensional wire-frame models represent a graphics object by connected polygons or faces. The model is defined by specifying the coordinates of the vertices of each polygon and then specifying the faces by connecting the specified vertices in a specific order. This three-dimensional modeling was based on a MATLAB function called "patch" (see "findInputSet.m" in the Appendix A).

Five ship types were chosen to be included in the recognition class and therefore be modeled: namely, an aircraft carrier, a frigate, a destroyer, a research ship (Point Sur), and a merchant ship. With these five types, it was possible to address a typical scenario at sea, where we can find military ships, small civilian ships and big merchant ships.

The three-dimensional wireframe model for the aircraft carrier was based on a 1:1800 scaled drawing of the Carl Vinson aircraft carrier (Nimitz Class) [Ref. 30]. This drawing is shown in Figure 4 and a picture of the Carl Vinson aircraft carrier is shown in

Figure 5. The model was implemented manually since no CAD model was available. It was formed of 45 vertices and 34 planes (see the "aircraft.m" program in Appendix A). Figure 6 shows the model from four view angles.
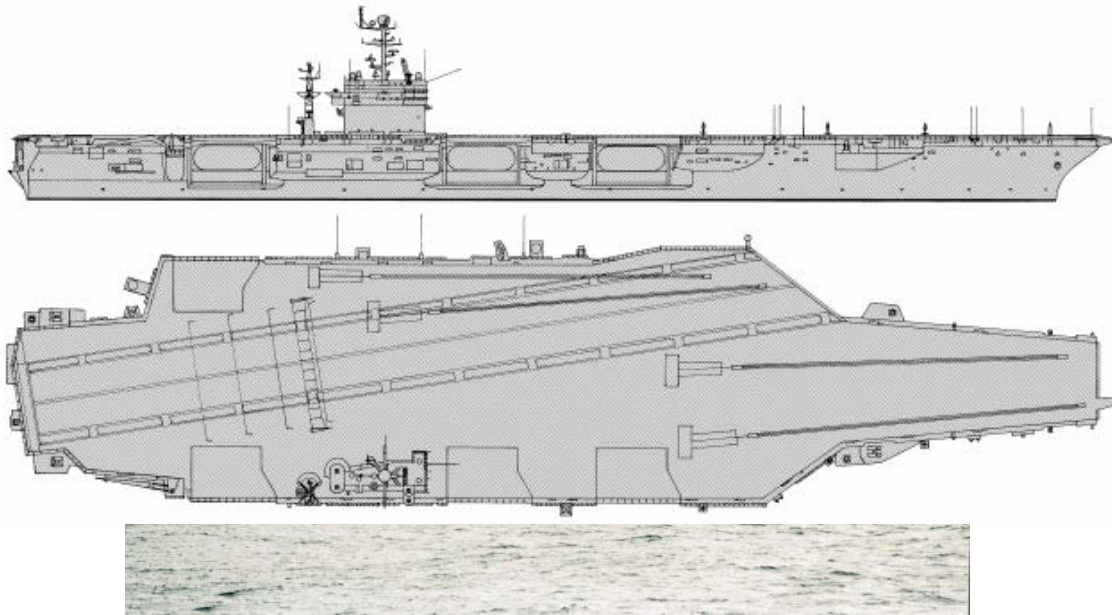
Figure 4: Scaled drawing of the Carl Vinson aircraft carrier [From Ref.30].

Figure 5: Picture of the Carl Vinson aircraft carrier [From Ref. 30].
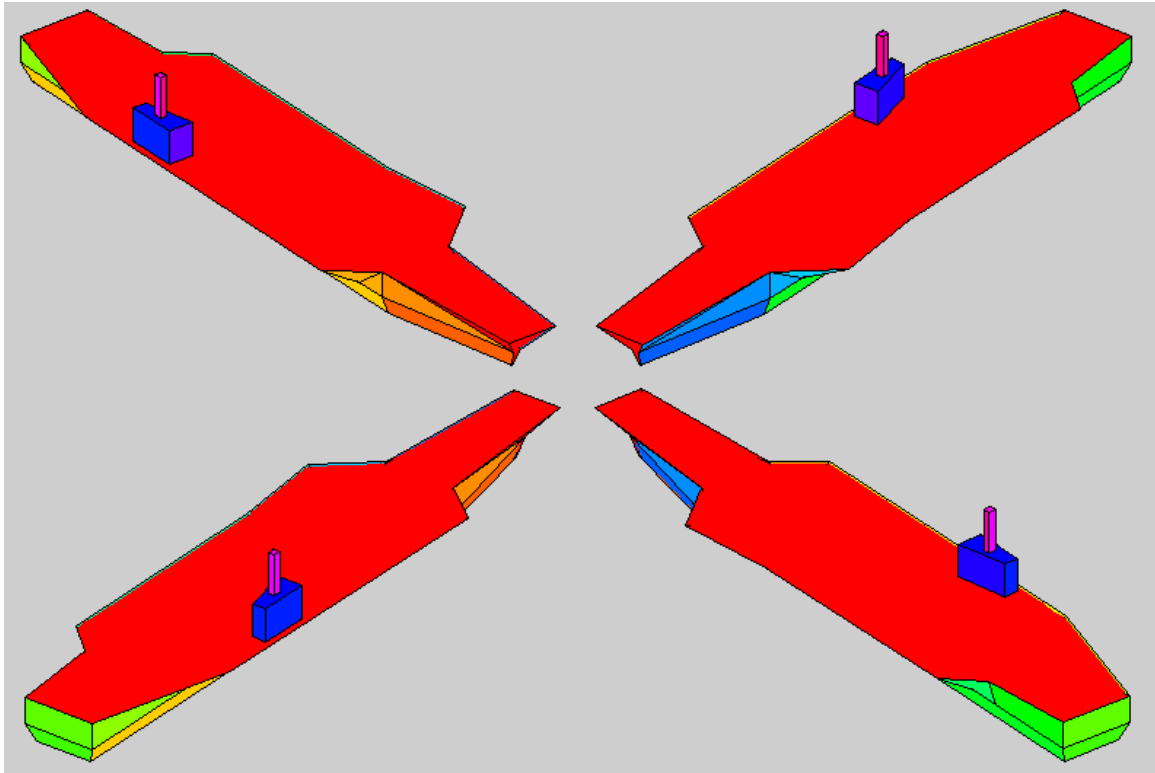
Figure 6: The aircraft carrier three-dimensional model in four view angles.
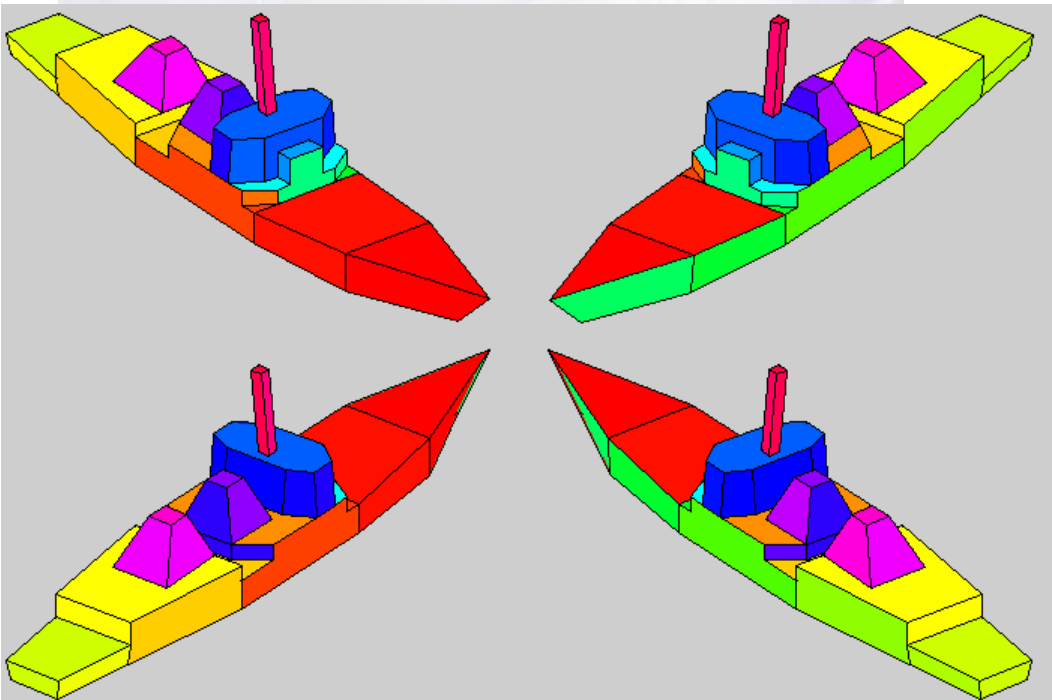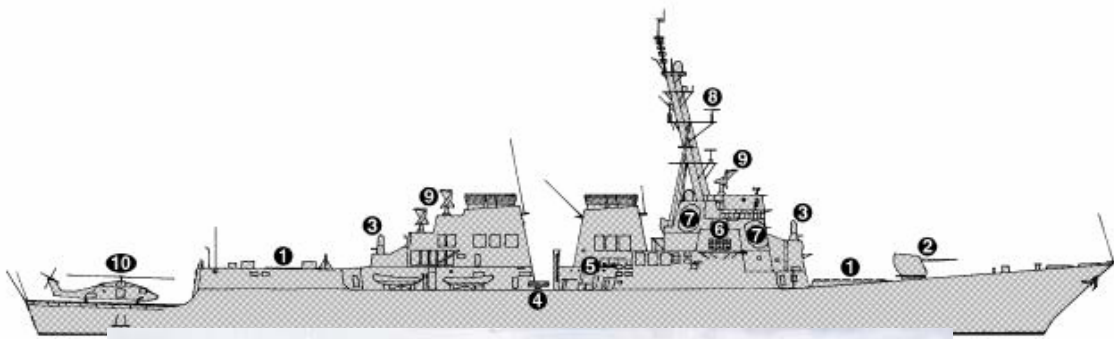
Figure 9: The destroyer three-dimensional model in four view angles.

The three-dimensional wireframe model for the frigate was based on a 1:1200 scaled drawing of the Rentz frigate (Oliver Hazard Perry Class) [Ref. 30]. This drawing is shown in Figure 10 and a picture of the Oscar Austin destroyer is shown in Figure 11. The frigate model was formed of 130 vertices and 66 planes (see the "frigate.m" program in Appendix A). Figure 12 shows the model from four different view angles.

Figure 10: Scaled drawing of the Rentz frigate [From Ref.30].

Figure 11: Picture of the Rentz frigate [From Ref.30].
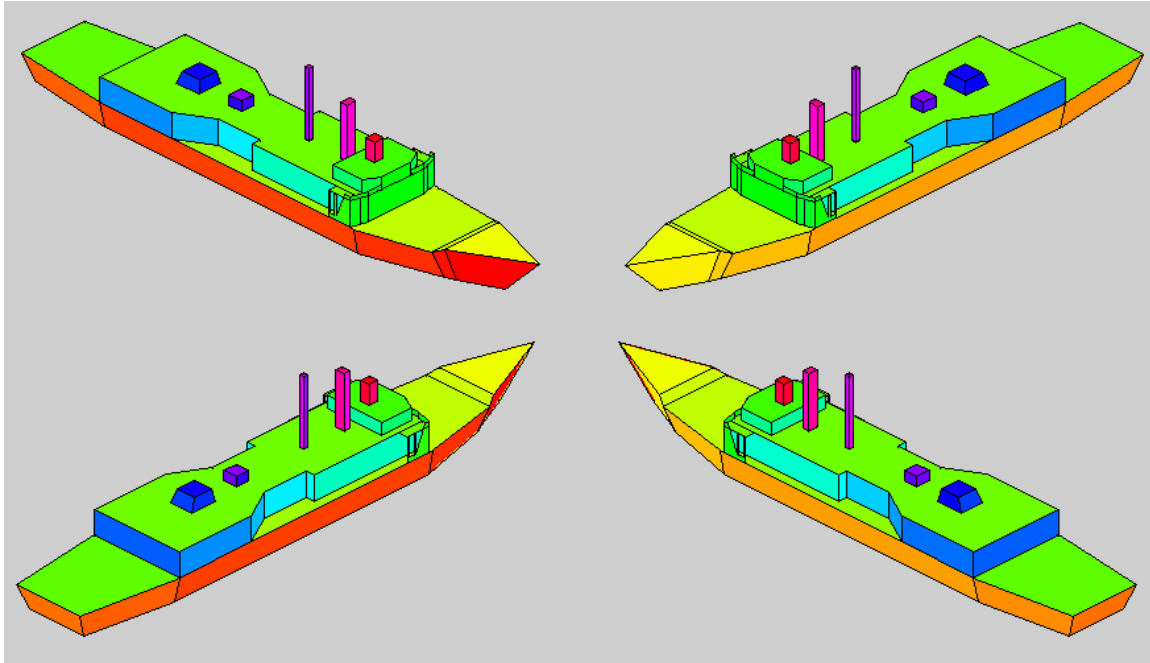
Figure 12: The frigate three-dimensional model in four view angles.
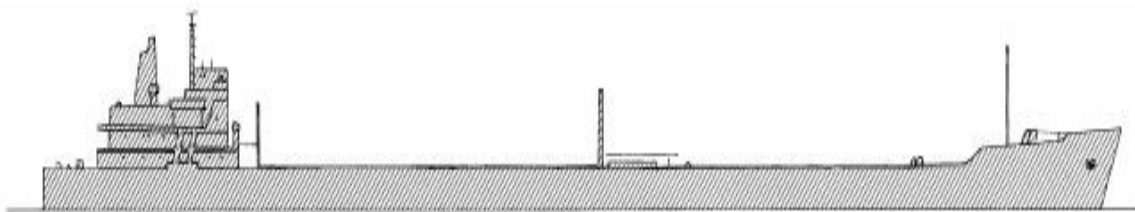
The three-dimensional wireframe model for the Merchant Ship was based on a scaled drawing of the Sea Isle City U.S. Tanker [Ref. 31]. This drawing is shown in Figure 13 and a picture of the Sea Isle City U.S. tanker is shown in Figure 14. The merchant model was formed of 100 vertices and 58 planes (see the "merchant.m" program in Appendix A). Figure 15 shows the model viewed from four view angles.

Figure 13: Scaled drawing of the Sea Isle City tanker [From Ref.31].

Figure 14: Picture of the Sea Isle City tanker [From Ref.31].
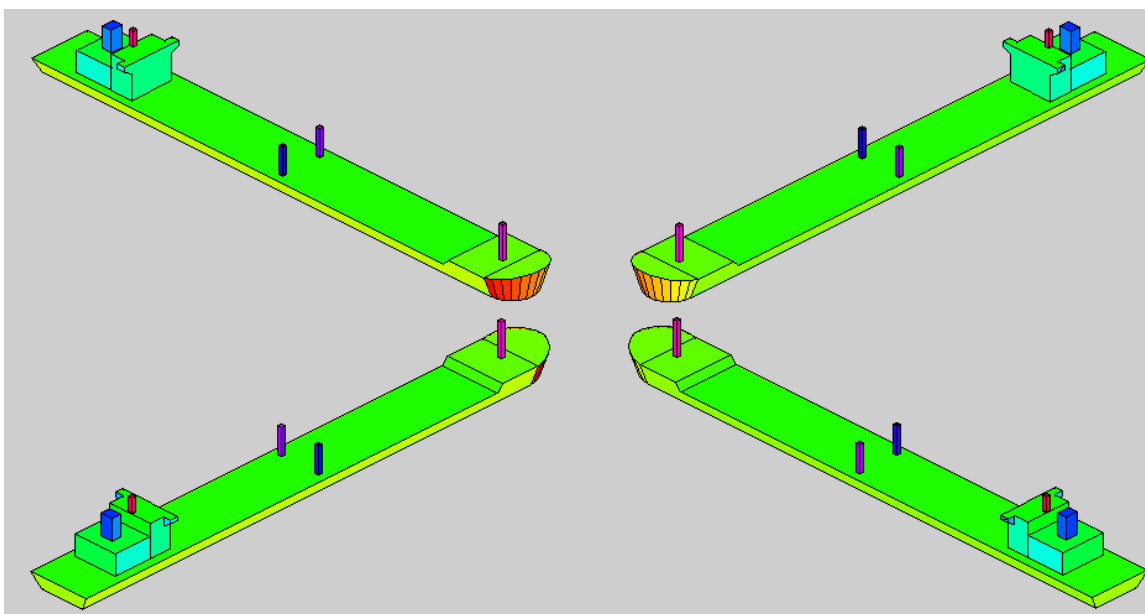
Figure 15: The merchant three-dimensional model in four different view angles.
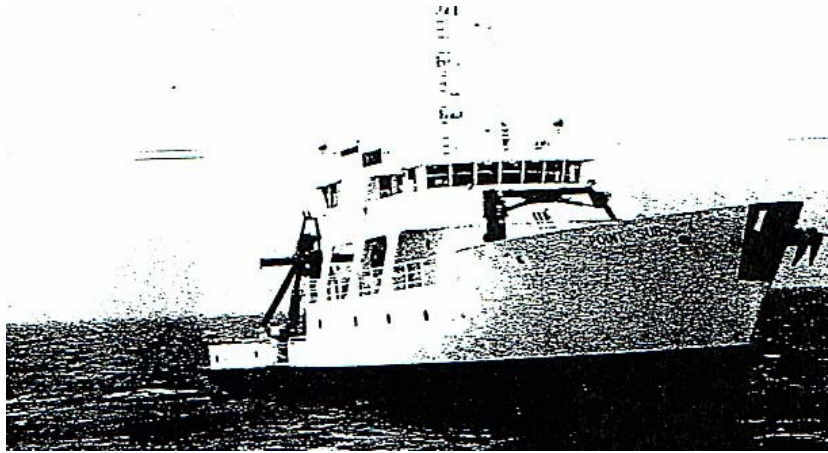
The three-dimensional wireframe model for the research ship was based on general specifications and dimensions extracted from R/V Point Sur Cruise Planning Manual (see Figure 16). The research ship model was formed of 76 vertices and 32 planes (see the "poitsur.m" program in Appendix A). Figure 17 shows the model viewed from four view angles.
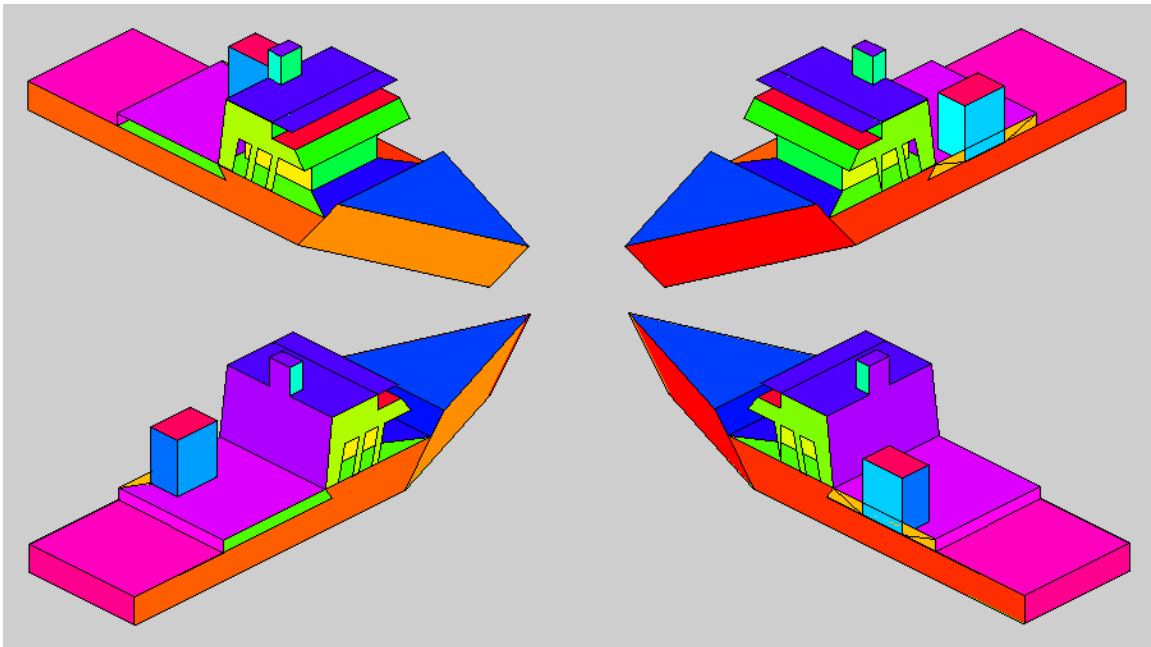
Figure 16: Picture of the R/V Point Sur.

Figure 17: The Point Sur three-dimensional model in four view angles.

2. Viewpoint Control
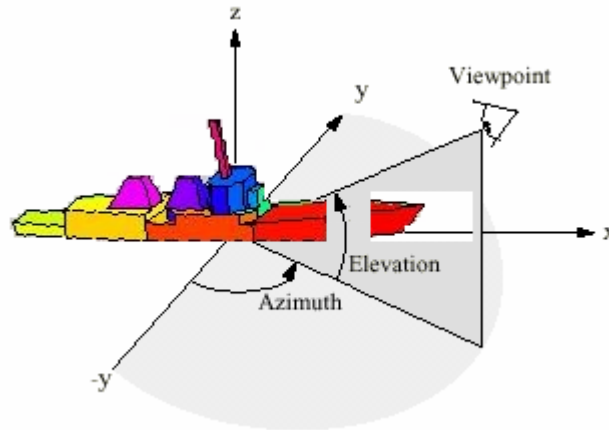
To extract silhouettes, the orientation of the three-dimensional ship model must be specified. It was possible to specify the viewpoint with the MATLAB "view" command by defining azimuth and elevation with respect to the axis origin. Azimuth is a polar angle in the x-y plane, with positive angles indicating counter-clockwise rotation of the viewpoint. Elevation is the angle above (positive angle) or below (negative angle) the x-y plane. The counter-clockwise concept for the azimuth was adopted because the viewing azimuth is the negative of the ship's heading. Therefore, a ship with heading of 30^0 clockwise is equivalent to viewing that ship with azimuth of 30^0 counter-clockwise. The diagram in Figure 18 illustrates the coordinate system. The arrows indicate positive directions. The origin was assumed to be located approximately in the center of gravity of the ship model. Only the portion above sea level was considered. Using this coordinate system, we can verify that the broadside view of any ship model corresponds to 0^0 in azimuth and 0^0 in elevation, in this situation the bow direction will be to the right (positive x).

Figure 18: Diagram illustrating the coordinate system and the ship model origin.

One view of each of the five modeled ships is illustrated in Figure 19. In this figure, the azimuth angle is –37.5 degrees and the elevation angle is 30 degrees.
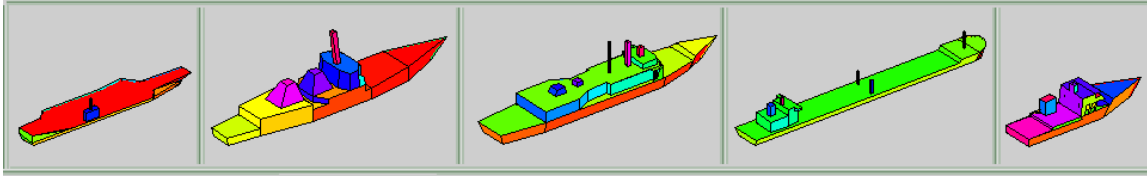
Figure 19: Views of the three-dimensional ship models (az.=-37.5^o, elev.=30°).

3.   Orthographic Projection
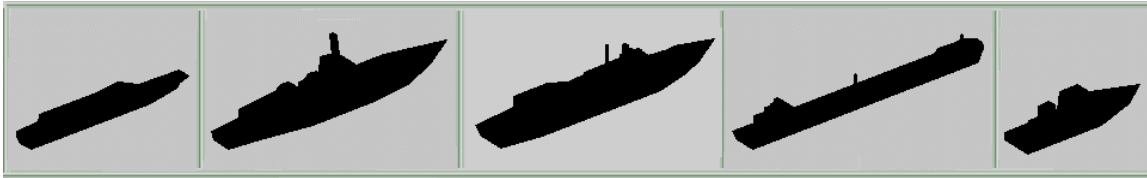
Once the three-dimensional model is created and the aspect angle is set using the "view" command, a silhouette can be created by projecting the three-dimensional ship model. The orthographic method projects the viewing volume as a rectangular parallelepiped onto a plane, i.e., relative distance from the camera does not affect the size of objects. Using orthographic projection it is possible to get a good approximation of the real process of image generation when the distance from the object to the camera is much greater than the relative deep of the object structural points. This applies to our task because ships are generally far away from the FLIR sensors. Figure 20 shows some silhouettes created using the orthographic projection applied to the images shown in Figure 19.

Figure 20: Silhouettes created using the images of Figure 19.

C.   THE REAL FLIR IMAGES DATABASE

1.   Domain Issues

We also obtained real images taken at sea using the AN/AAS-44V Forward-Looking Infrared (FLIR) sensor, mounted on a springboard at the nose of the SH-60B

Rapid Deployment Kit equipped helicopter. The FLIR images were available through a VHS-format videotape showing several ships. However, only images of our modeled ships were considered for our analysis. The FLIR images show good contrast and were displayed in black-hot (higher temperatures areas are black) format. Image frames were acquired using a commercial video-grabber board installed in a PC-type desktop computer.

The real FLIR images that we extracted from the videotapes were used to test our recognition system, previously trained with the three-dimensional ship model data. This was considered particularly important since presenting the classifier with new images with blurriness and an unknown target viewpoint is challenging.
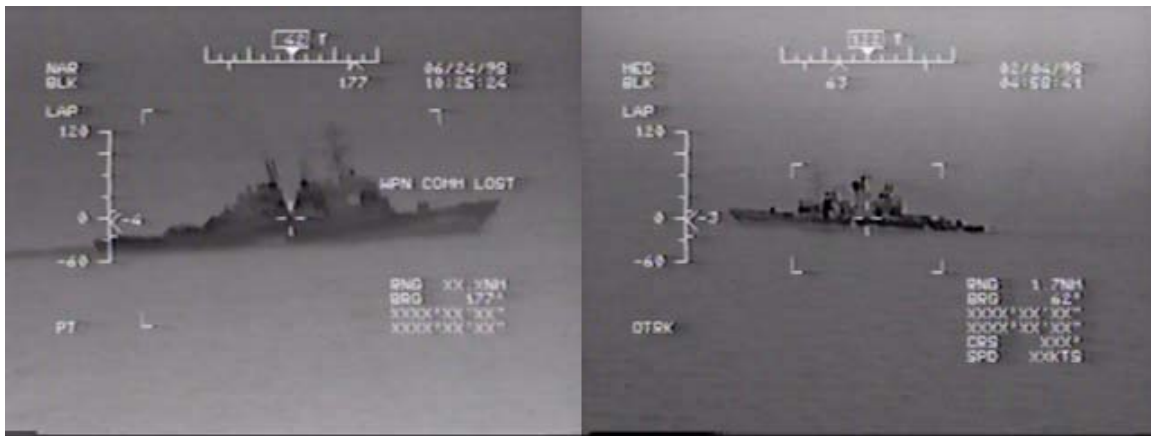
Only 25 real FLIR images were used for testing due to the small number of modeled ships in the FLIR tape. These were: two destroyer images (Figure 21), four aircraft carrier images (Figure 22), 15 merchant ship images (Figure 23), four research ship images (Figure 24), and no frigate images.

Figure 21: Real FLIR images of destroyers
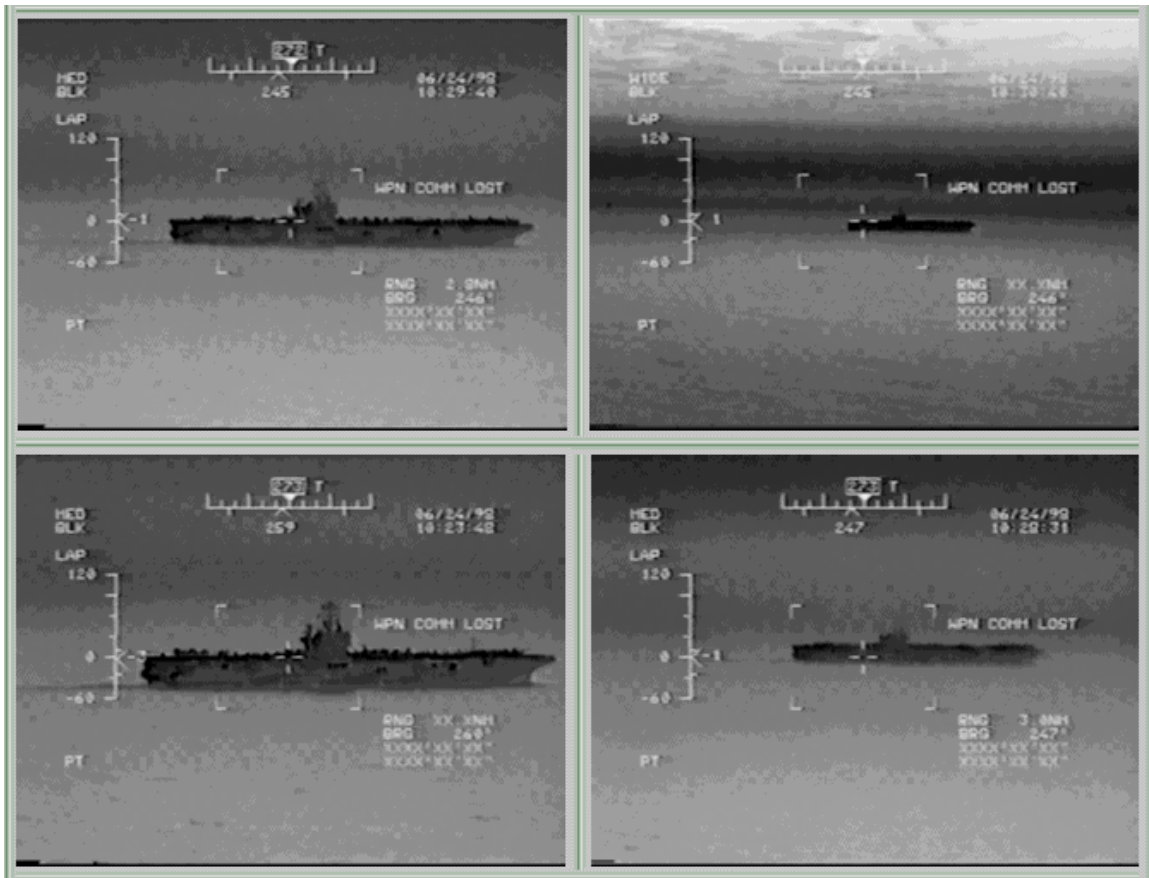
Figure 22: Real FLIR images of aircraft carrier class.
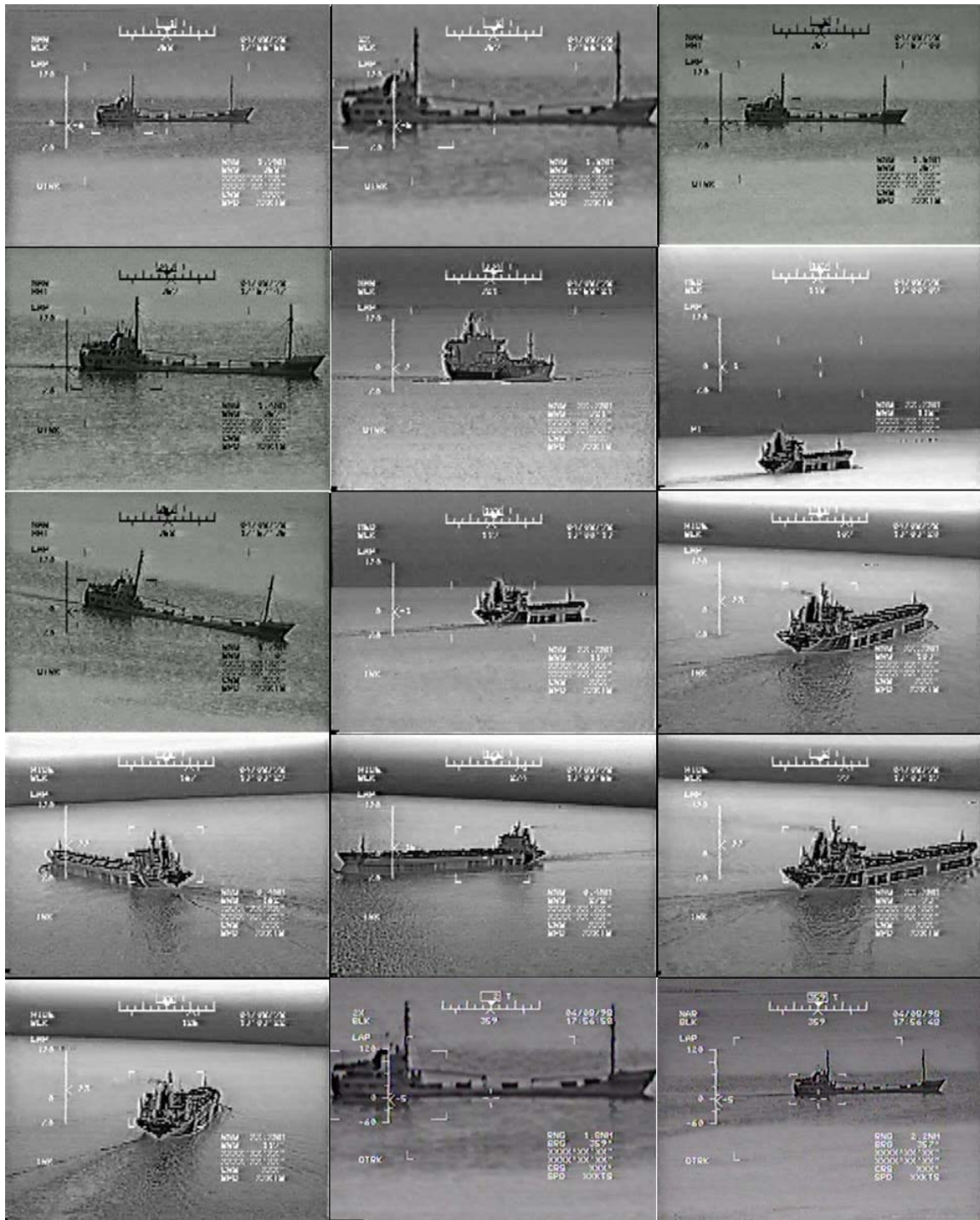
Figure 23: Real FLIR images of merchant ship class.

Figure 24: Real FLIR images of research ship class.

The quality of the images was not ideal because the FLIR system projects alphanumeric data and targeting aids onto the screen. Figure 25 shows one of the FLIR images. Specifically, the crosshairs partially obscure the ship image and interfere with the classification process. A segmentation was necessary to eliminate the background including the alphanumeric data.

Figure 25: FLIR image of the Arleigh Burke Destroyer.

2. Segmentation

Prior to computing the moment invariants of the ship of the Figure 25, we must suppress the background and extract the ship silhouette. In this segmentation, we employed histogram and thresholding techniques.

We assume that the extracted 320x240 pixels image contains one ship only. This image includes the ship, water, alphanumeric data, and may include the sky. As the first step, we generated the gray-level histogram of the image and selected a threshold level that best extracted the ship from the water region. Figure 26 shows the histogram of Figure 25.
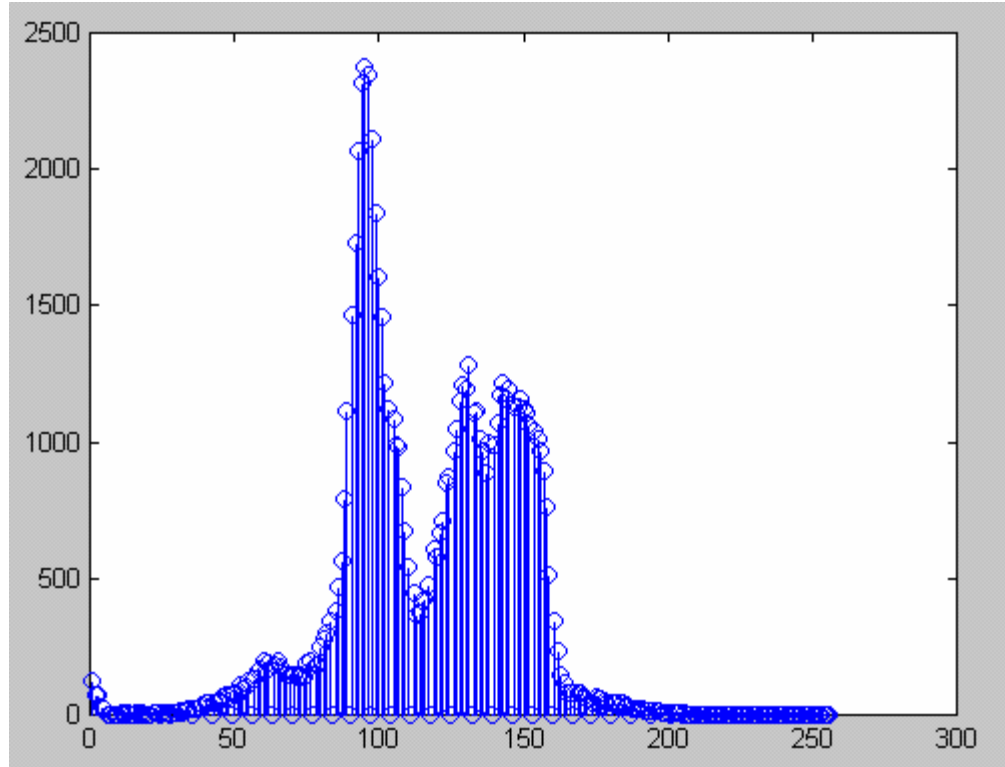
Figure 26: Gray-level histogram of Figure 21.

In this image, the average brightness for the water region is greater than that for the ship region. If we had a sky region in this picture, the values in the sky region would be lower than in the ship region. The histogram profile was analyzed and it was verified that the highest peak was related to the water portion of the image. It was also verified that the first peak left of the highest was related to the ship. The region between the "ship peak" and the "water peak" was a transition region; the threshold value selected corresponded to the minimum in the transition region. In Figure 26, those values are water peak= 95, ship peak= 65 and threshold= 72. The original image was thresholded at that value and a binary image generated. Figure 27 shows the result for Figure 25.

29

Figure 27: Binary image thresholded at level 72 using Figure 25.
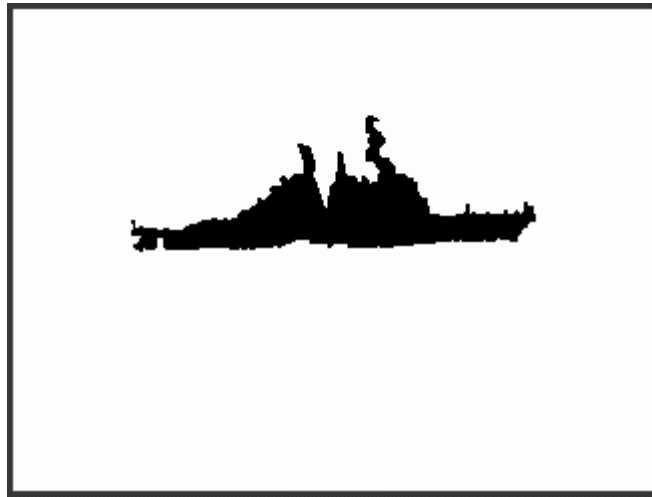
In a second step, we eliminated spurious pixels by extracting the greatest connected region and filling the holes (see segmentation.m in Appendix A). The final ship silhouette found for Figure 25 is shown if Figure 28.

Figure 28: Silhouette found after cleaning up Figure 27.

D.   TRAINING PHASE OF THE NEURAL NETWORK CLASSIFIER

The neural network ship classifier required a training phase using representative projective views. As explained in Section B of Chapter II, the moment functions

invariance under reflection helps to simplify the range of all representative views distinct ship views. Three-dimensional objects which possess symmetry about a plane, such as a ship, can have its significant range of distinct views for azimuth restricted to [-90°, 90°], where 0° corresponds to the broadside ship silhouette [Ref. 16]. Elevation angles were restricted to the upper hemisphere with 45 degrees as the upper operational limit, as the helicopter will be viewing the ship with lower elevation angles.

One training set and two testing sets of projections were generated. The training set was 48 views of each of the five ship types taken at viewpoints separated by 15° in a 180°x45° sector; i.e., { (θ, α), θ = -90°, -75°, -60°, …, 75°; α = 0°, 15°, 30°, 45° } where θ and α represent azimuth and elevation angles respectively. Examples of the training images are in Figure 29 below. In this figure, the elevation angle is 15° and the azimuth angles from left to right are: -90°, -60°, -30°, 0°, 30°, and 60°. The ship types from top to bottom are aircraft carrier, destroyer, frigate, merchant, and research ship.

Figure 29: Examples of the training images for each modeled ship

31

E.   TESTING PHASE

There were two testing sets; 41400 silhouettes projected from the three-dimensional ship models and the 25 real FLIR images. The first test set contains 8280 views of each of the five ship models taken at viewpoints separated by 1° in azimuth and 1° in elevation; i.e., $\{(\theta,\alpha), \theta = -90^o, -89^o, -88^o, \ldots, 89^o ; \alpha = 0^o, 1^o, 2^o, \ldots, 45^o \}$ where θ and α represent azimuth and elevation angles respectively. Although this set contains the training set, the number of training views was very small (96 views) to compromise the simulation results. The real image test set was described earlier.

F.   PROGRAMS DEVELOPED

All 16 programs were written in MATLAB. They can be divided in four categories (see Table 1 and Appendix):

- Three-dimensional ship modeling: specialized functions used to create the three-dimensional models used in other programs;

- Moment invariant computation: programs used to calculate the moment invariants of a specific ship silhouette;

- Neural network training: programs used to train the neural network implemented; and

- Testing: programs used to evaluate the performance of the system implemented.

Name	Type	Function
aircarrier.m	Three-dimensional ship modeling programs	Create the model for aircraft carrier
destroyer.m		Create the model for destroyer
frigate.m		Create the model for frigate
pointsur.m		Create the model for research ship
merchant.m		Create the model for merchant ship
findInputMomSet.m	Moment invariants computation programs	Returns the 12-element input set
find_mom_functions.m		Returns the six moment functions values
find_moment.m		Returns the central moment
find_centroid.m		Returns the centroid of a silhouette
mainShipRecon.m	Neural network training programs	Creates and trains a neural network responsible for recognizing ship types
findInputSet.m		Returns all the silhouettes to be used by the neural network during training phase

interface.m	Testing programs	Create a Graphical User Interface to evaluate the system implemented. Three ship silhouettes are shown in the interface: (1)the original silhouette, (2)the rotated, scaled and noisy silhouette defined by the user, and (3)the neural network guessed silhouette
plotSilhouette.m		Draws the ship silhouette inside the Graphical User Interface
segmentation.m		Segments a FLIR real image using a histogram and threshold technique
createTestSet.m		Creates 05 mat files containing the silhouettes of each ship for increments of one degree in azimuth and elevation, then plots the errors
findResultSet.m		Returns a vector with the size of all the viewangles being tested, where "1" will mean misclassified and "0" will mean correct classified

Table 1: Programs implemented

THIS PAGE INTENTIONALLY LEFT BLANK

V. RESULTS FROM EXPERIMENTATION

A. EVALUATION STRUCTURE

Experiments were carried out in order to evaluate the proposed system. For this purpose, the five ship models detailed in chapter IV were used. The parameters generally used to characterize the overall performance of an automatic target recognition system are the probability a given ship is correctly recognized (recall) and probability a given type identification is correct (precision). By classification is meant recognition, the determination of the target type. Performance data is generally given in the form of a confusion matrix together with the size of the feature vector and database.

Using the probability of correct classification as a reference and the simulated three-dimensional ship model images as a database, it was possible to iteratively optimize our system. This optimization was achieved in three distinct experiments.

B. FIRST EXPERIMENT

The first experiment was implemented as described in chapter IV. Experiments were performed with simulated 12-components moment invariants signatures from the models. The six moment invariants of the solid silhouette and the six moment invariants of the boundary make up the signature vector, as described in chapter II. The training signatures were generated from images of the three-dimensional models taken at a regular pace of 15 degrees in azimuth increments and in four different viewing elevation angles: 0°, 15°, 30° and 45°. The total training set was 48 images.

The testing set contained the images of the three-dimensional models taken at a regular pace of 1° in azimuth increments and using the same four elevation angles used during training. The neural network used was small, with only 20 hidden neurons and a single hidden layer. Backpropagation was used as the training technique. The network was successfully trained. This network yielded 90.1% discrimination leaving an overall approximate 10% error rate. Details of the experiment are summarized in Table 2.

Network Parameters	Network-1
# input nodes	12
# nodes in hidden layer	20
# output nodes	5
# training set	240 (12 x 4 x 5)
# test set	3600 (180 x 4 x 5)
Accuracy	90.1%

Table 2: Neural network for experiment 1
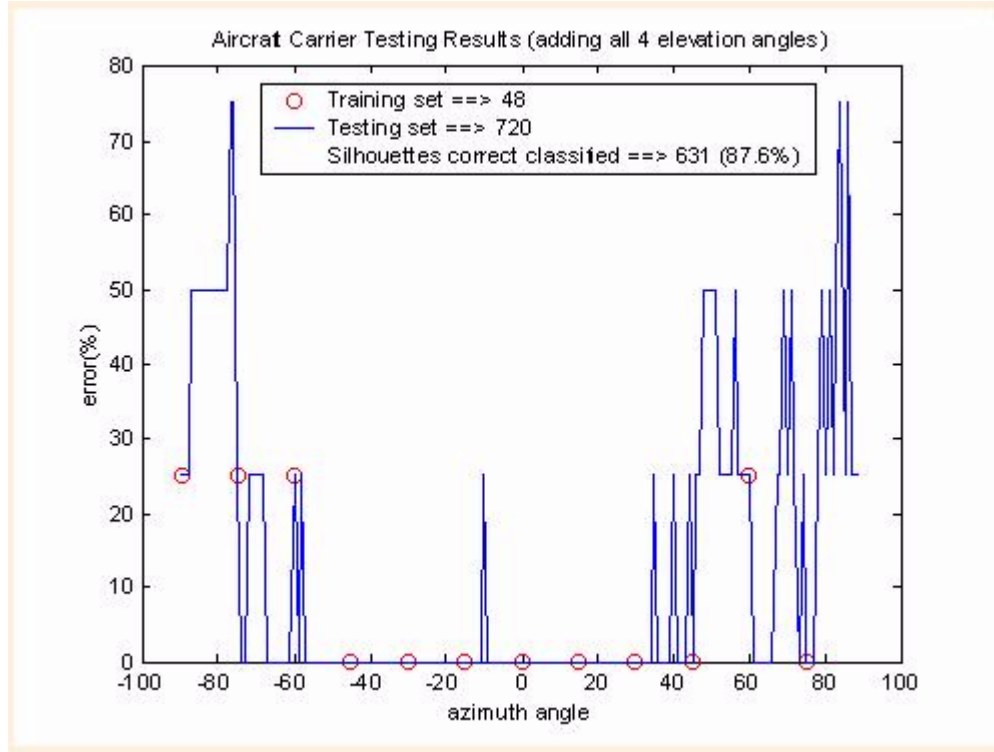
The first experiment enabled us to analyze the azimuth behavior of the neural net generalization capability. For this purpose, we plotted for each ship type the classification error percentage with respect to azimuth (Figures 30 to 32).

Figure 30: Accuracy with respect to azimuth for aircraft carrier
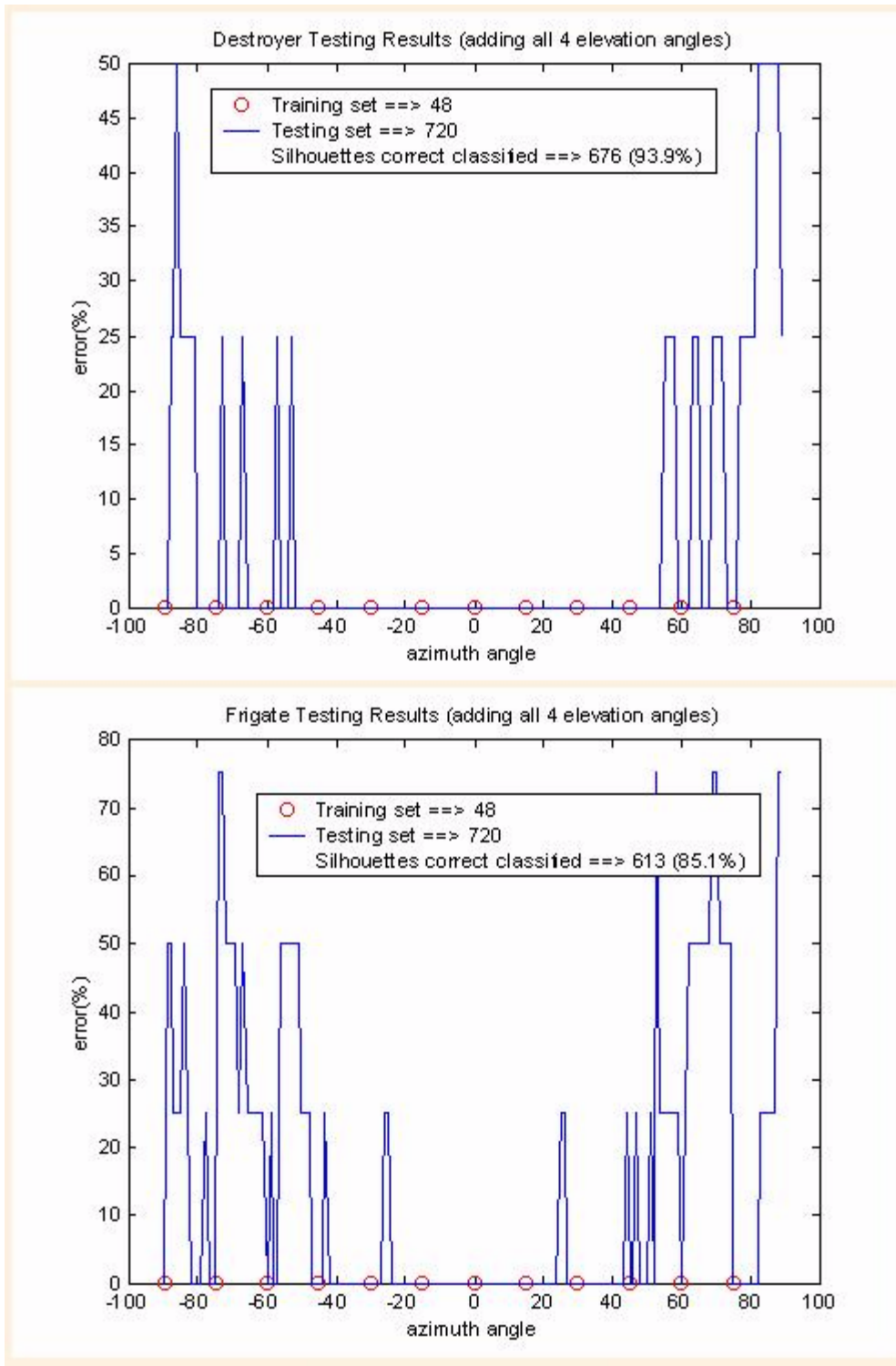
Figure 31: Accuracy with respect to azimuth variation for destroyer and frigate
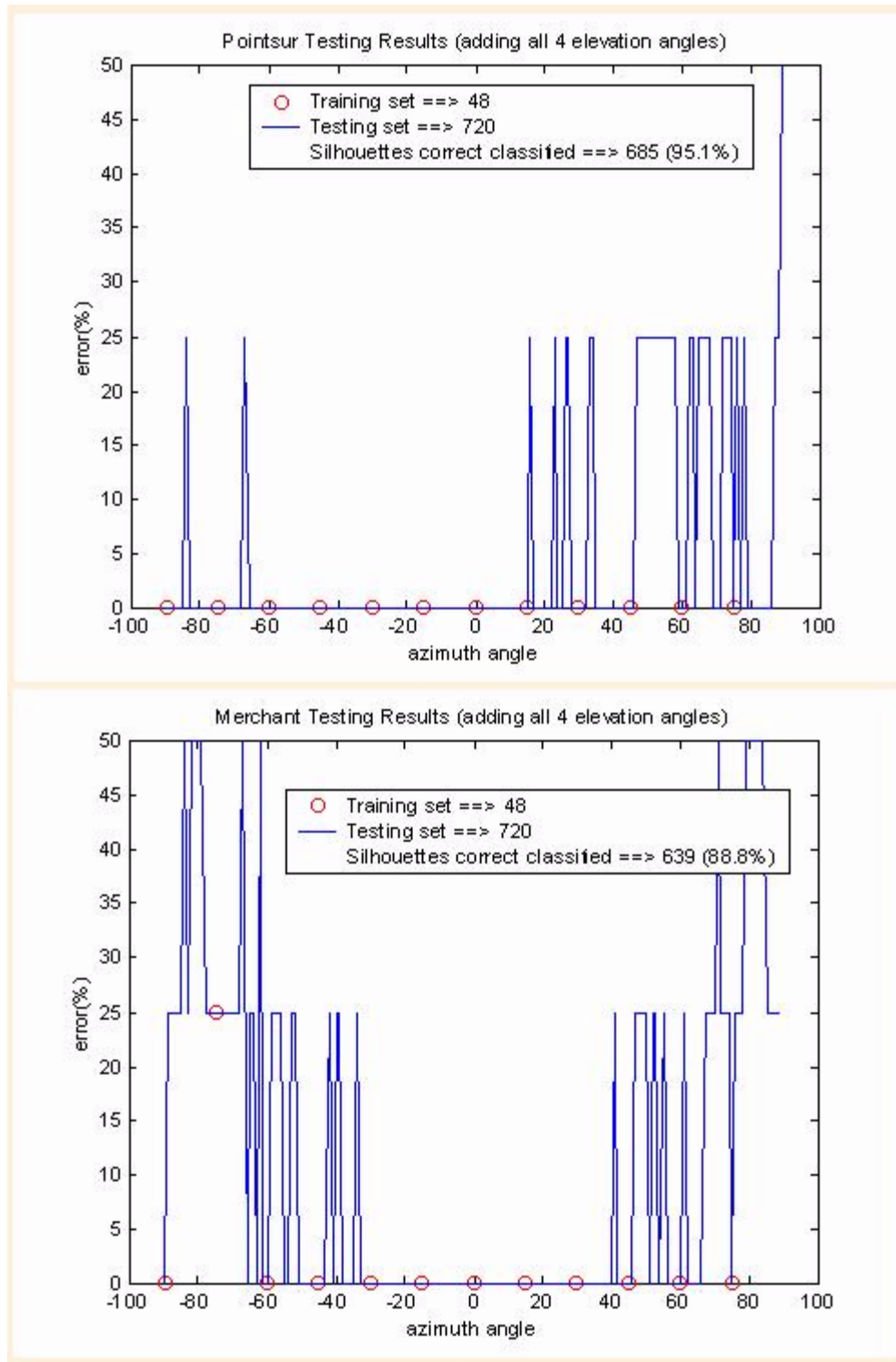
Figure 32: Accuracy with respect to azimuth variation for Point Sur and merchant ship

Analyzing the results of these graphs, we can verify that the major errors were related to high azimuth angles. This error pattern helped us to decide to use an unevenly spaced training set. In the second experiment, we increased the number of azimuth training angles by using small steps (5°) in high azimuth angles.

C. SECOND EXPERIMENT

In the second experiment, we increased the number of training azimuth angles by using an unevenly spaced training set. For low azimuth angles (-45° to 45°), we kept the 15° step, but for high azimuth angles we chose a smaller step of 5° .The new training set contained 96 views of each of the five ship types taken at the following viewpoints: {(θ, α), θ = -90, -85, -80, -75, -70, -65, -60, -55, -50, -45, -30, -15, 0, 15, 30, 45, 50, 55, 60, 65, 70, 75, 80, 85; α = 0, 15, 30, 45}, where θ and α represent azimuth and elevation angles respectively.

The neural network and testing set were the same used in the first experiment. We obtained 91.2% accuracy. We also tried another neural network architecture with 30 hidden nodes and the classification rate improved to 94.8%. Details of these experiments are summarized in Table 3.

Network Parameters	Network-2	Network-3
# input nodes	12	12
# nodes in hidden layer	20	30
# output nodes	5	5
# training set	480 (24 x 4 x 5)	480 (24 x 4 x 5)
# test set	3600 (180 x 4 x 5)	3600 (180 x 4 x 5)
Accuracy	91.2%	94.8%

Table 3: Neural networks for experiment 2

As we can see in Table 3, with a bigger training set and 30 hidden neurons the neural network-3 yielded the best accuracy. Figures 33 to 35 show the classification error percentage with respect to azimuth for network-3.
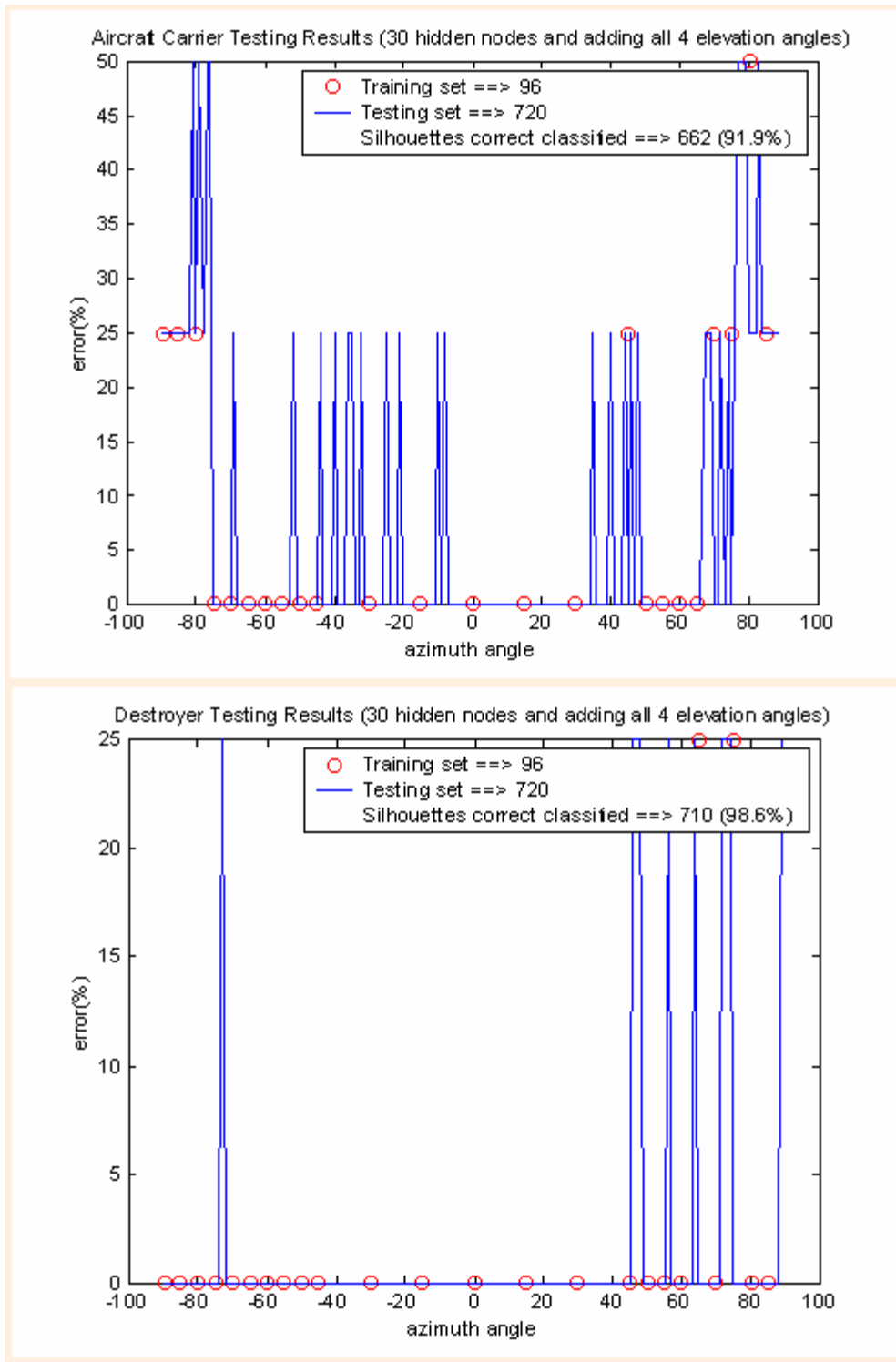
Figure 33: Accuracy with respect to azimuth for aircraft carrier and destroyer
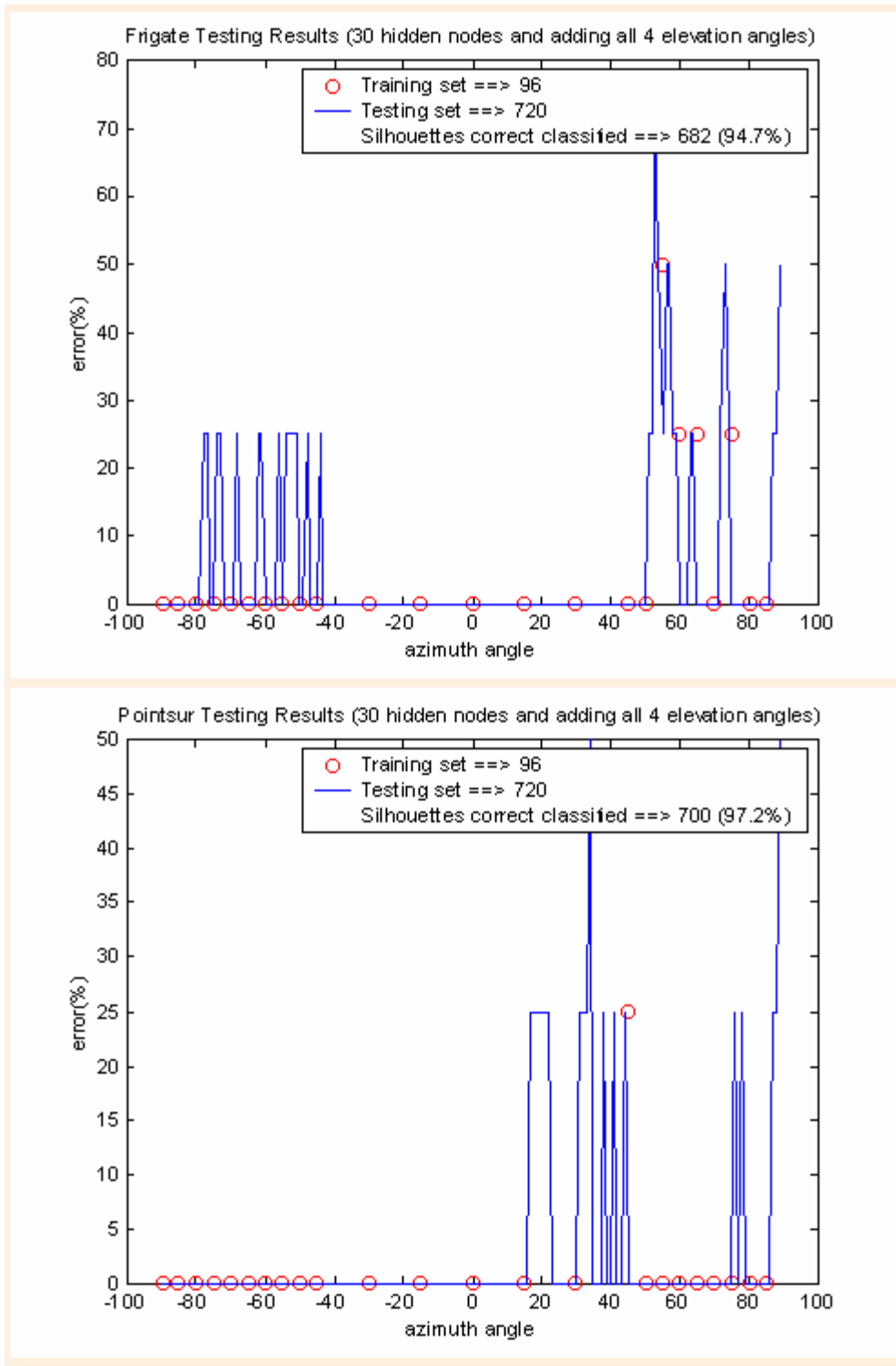
Figure 34: Accuracy with respect to azimuth for frigate and Point Sur
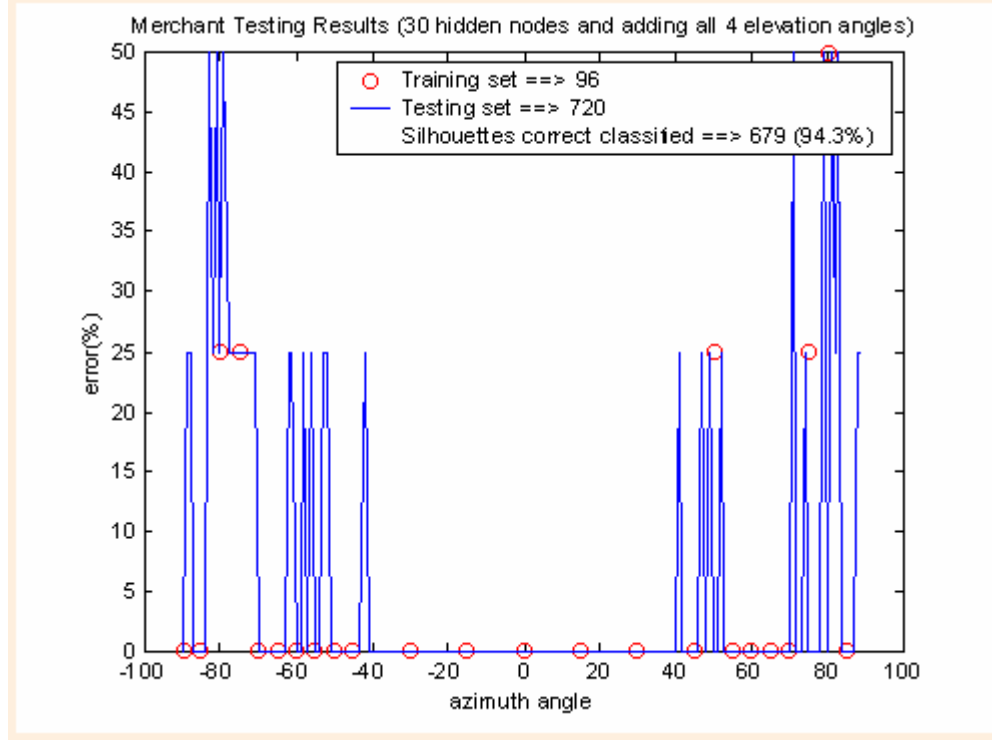
Figure 35: Accuracy with respect to azimuth for merchant ship

To analyze the elevation behavior of the neural network-3, we graphed, for each ship type, the classification error percentage with respect to elevation. Views of the three-dimensional model were calculated in increments of 1° in elevation, starting with 0° and ending with 45°, and using the same 24 azimuth angles used during training phase of network-3 (Figures 36 to 38).

Analyzing Figures 36 to 38, we can verify that the major errors were related to small elevation angles. This elevation error pattern inspired us to perform the third experiment, this time we increased the number of elevation training angles. The idea was to improve the classification accuracy of network-3 by training with more elevation angles.

44
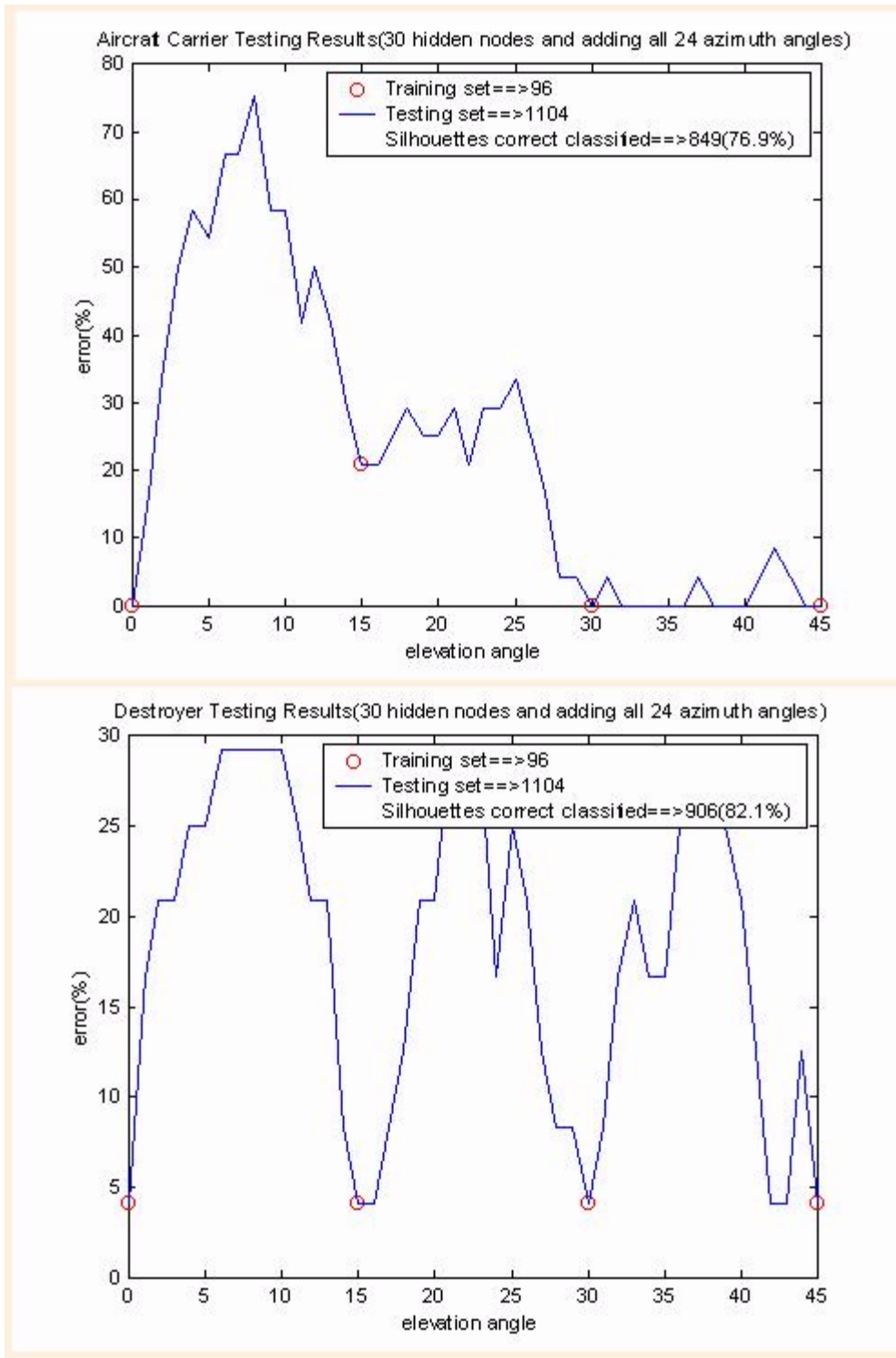
Figure 36: Accuracy with respect to elevation for aircraft carrier and destroyer
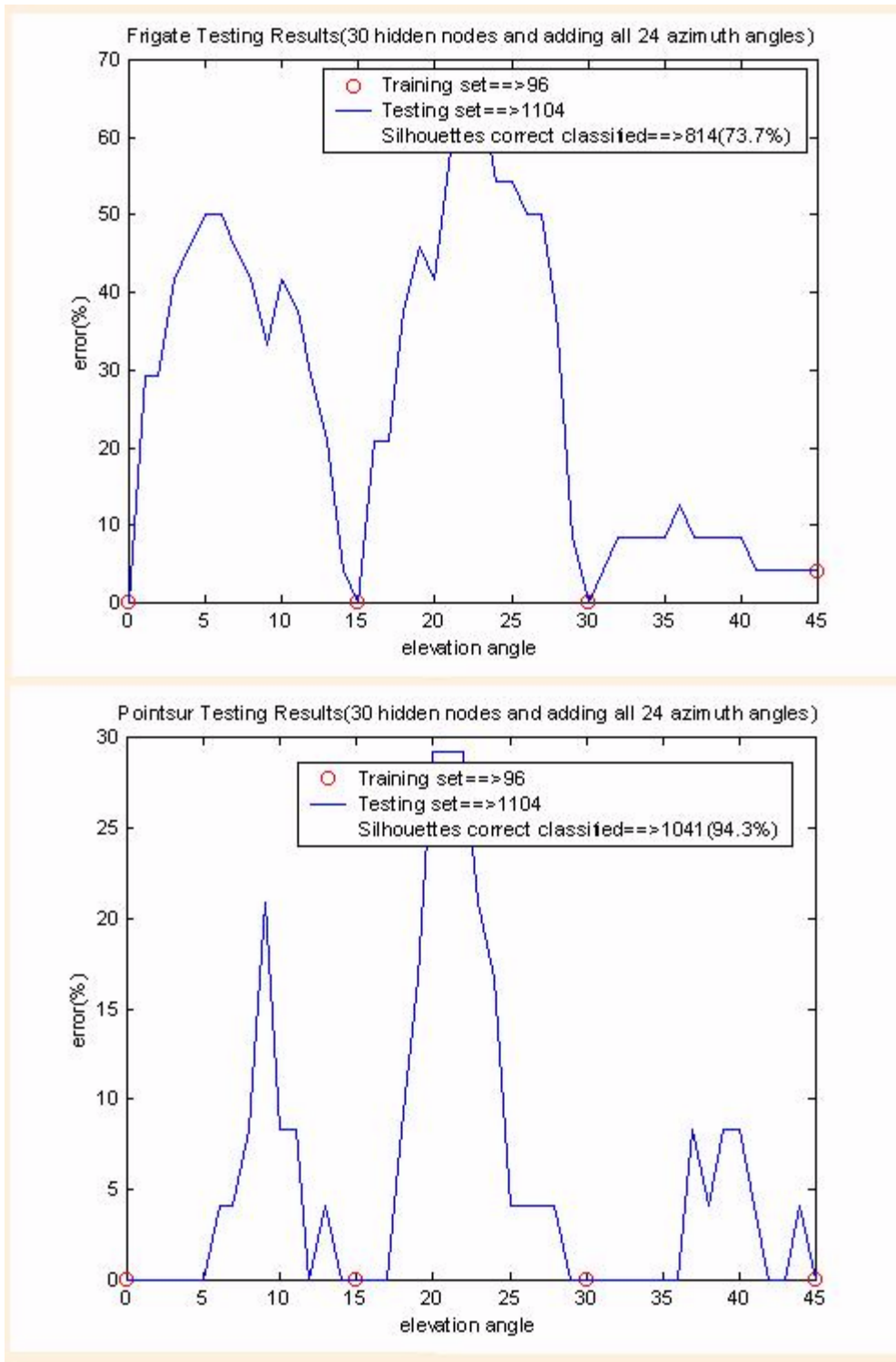
Figure 37:Accuracy with respect to elevation for frigate and Point Sur
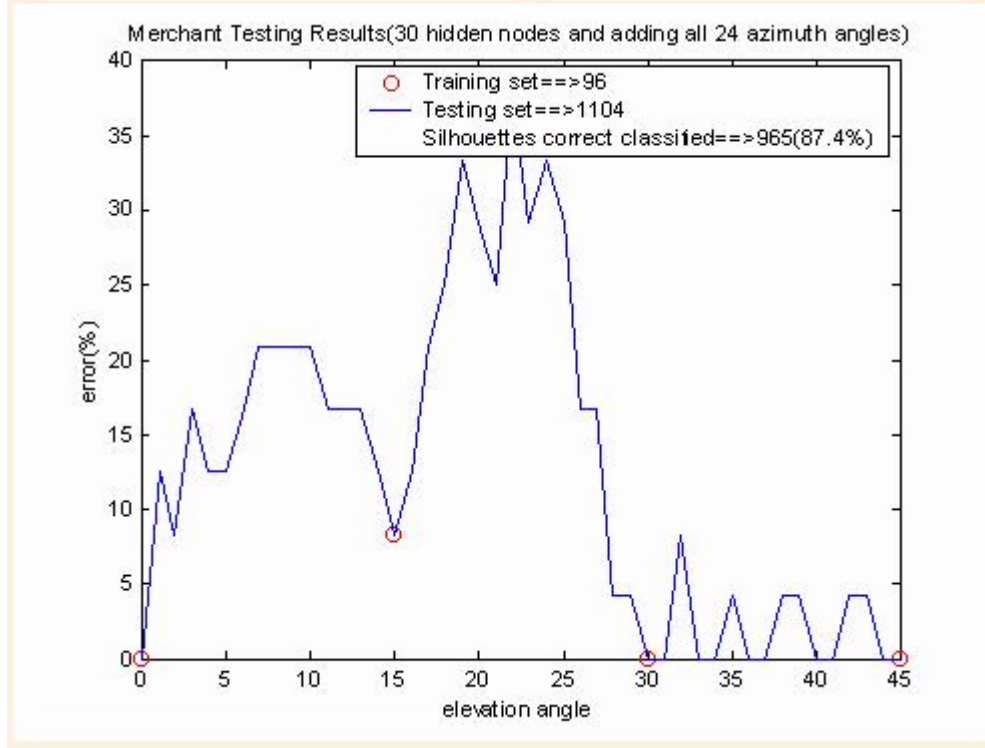
Figure 38:Accuracy with respect to elevation for merchant ship

D. THIRD EXPERIMENT

To improve the classification accuracy of network-3 we increased the number of elevation angles used for training in a third experiment. The new training set contains 168 views of each of the five ship types taken at the following viewpoints: $\{(\theta, \alpha), \theta = -90^o, -85^o, -80^o, -75^o, -70^o, -65^o, -60^o, -55^o, -50^o, -45^o, -30^o, -15^o, 0^o, 15^o, 30^o, 45^o, 50^o, 55^o, 60^o, 65^o, 70^o, 75^o, 80^o, 85^o; \alpha = 0^o, 7^o, 15^o, 22^o, 30^o, 45^o\}$, where θ and α represent azimuth and elevation angles respectively. The neural network and testing set were the same used before. We obtained 85.4% discrimination leaving an overall 14.6% error rate (Table 4).

This new architecture presented worse results because the neural network did not converge during the training phase using the backpropagation algorithm. This convergence problem was due to the large number of the training samples.

Network Parameters	Network-4
# input nodes	12
# nodess in hidden layer	30
# output nodes	5
# training set	720 (24 x 6 x 5)
# test set	3600 (180 x 4 x 5)
Accuracy	85.4%

Table 4: Neural network for experiment 3

E.   FOURTH EXPERIMENT

The third experiment has proved that the neural network-3 (accuracy of 94.8%) was our optimized solution. Thus, it was adopted for testing with the whole set of model images and with the real FLIR images.

The fourth experiment was to apply our approach to a large final testing set. This final set contained the images of the three-dimensional ship models taken at a regular pace of 1° in azimuth and elevation increments, totaling 41400 images (180 x 46 x 5). This experiment was performed using the neural network-3, from the second experiment. The average classification accuracy achieved was 87.3%. Table 5 shows the confusion matrix of counts.

Inferred Type / Input Type.	Aircraft Carrier	Destroyer	Frigate	Point Sur	Merchant	Accuracy
Aircraft Carrier	6711	201	425	393	550	81.1%
Destroyer	318	7301	397	257	7	88.2%
Frigate	345	788	6809	217	121	82.2%
Pointsur	67	146	177	7873	17	95.1%
Merchant	297	188	291	49	7455	90.0%
Precision	86.7%	84.7%	84.1%	89.6%	91.5%	
Overall probability of classification:						87.3%

Table 5: Confusion matrix for 41,400 views of modeled images and network-3

F.   FIFTH EXPERIMENT

We also ran the system on the 25 real FLIR images, as described in chapter IV. This experiment was also performed using the neural network-3, from the second experiment. The average classification accuracy achieved was 68%. Table 6 shows the confusion matrix of counts.

Inferred Type / Input Type	Aircraft Carrier	Destroyer	Point Sur	Merchant	Recall
Aircraft Carrier	3	0	0	1	75%
Destroyer	0	2	0	0	100%
Pointsur	0	1	3	0	75%
Merchant	6	0	0	9	60%
Precision	33.3%	66.6%	100%	90%	
Overall probability of classification:					68%

Table 6: Confusion matrix for real ship FLIR images and network-3.
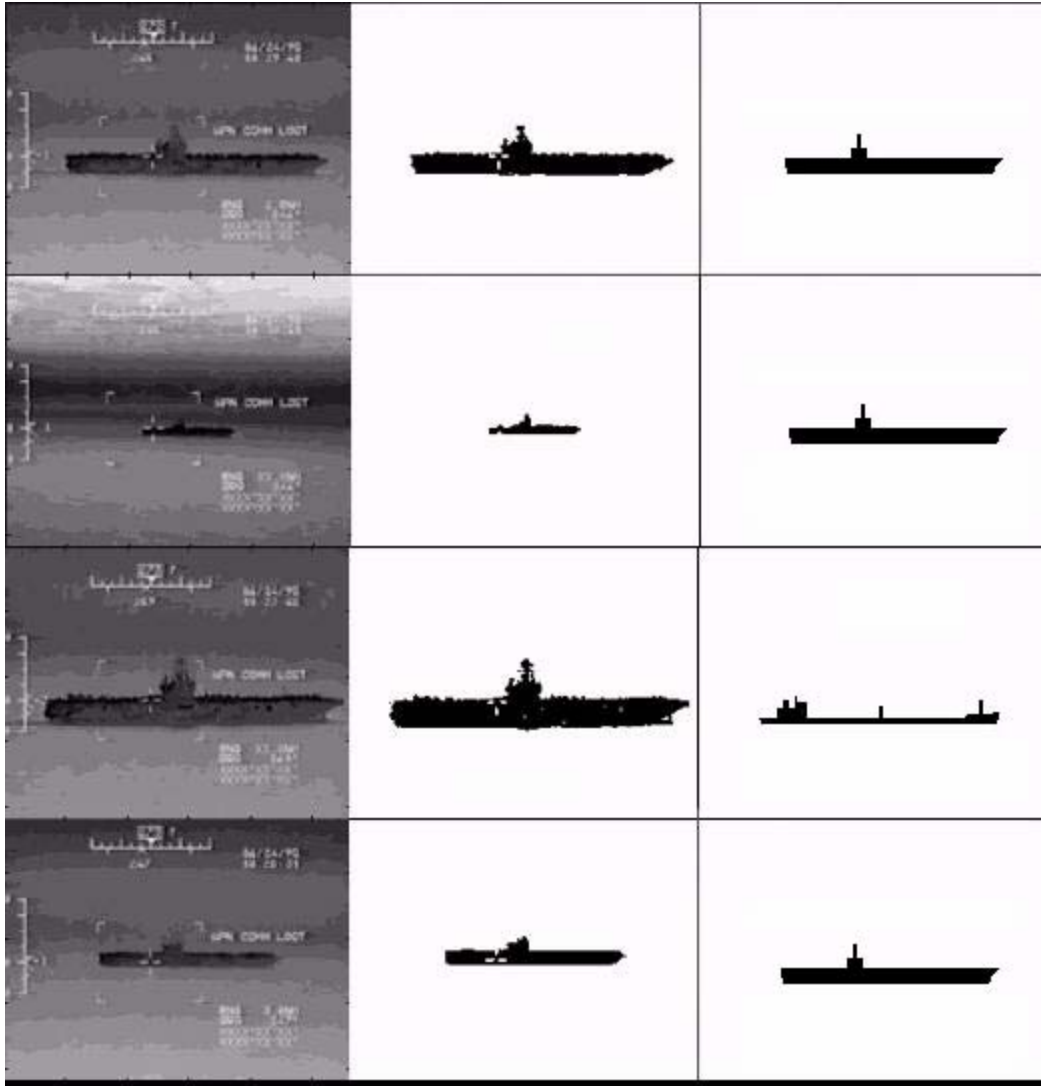
Figures 39 to 44 show all the FLIR images and respective results. The first columns show the original FLIR images; the middle columns show the silhouettes after the segmentation process; and the third columns show the neural network's guess.

Figure 39: Classification results for aircraft carrier FLIR images
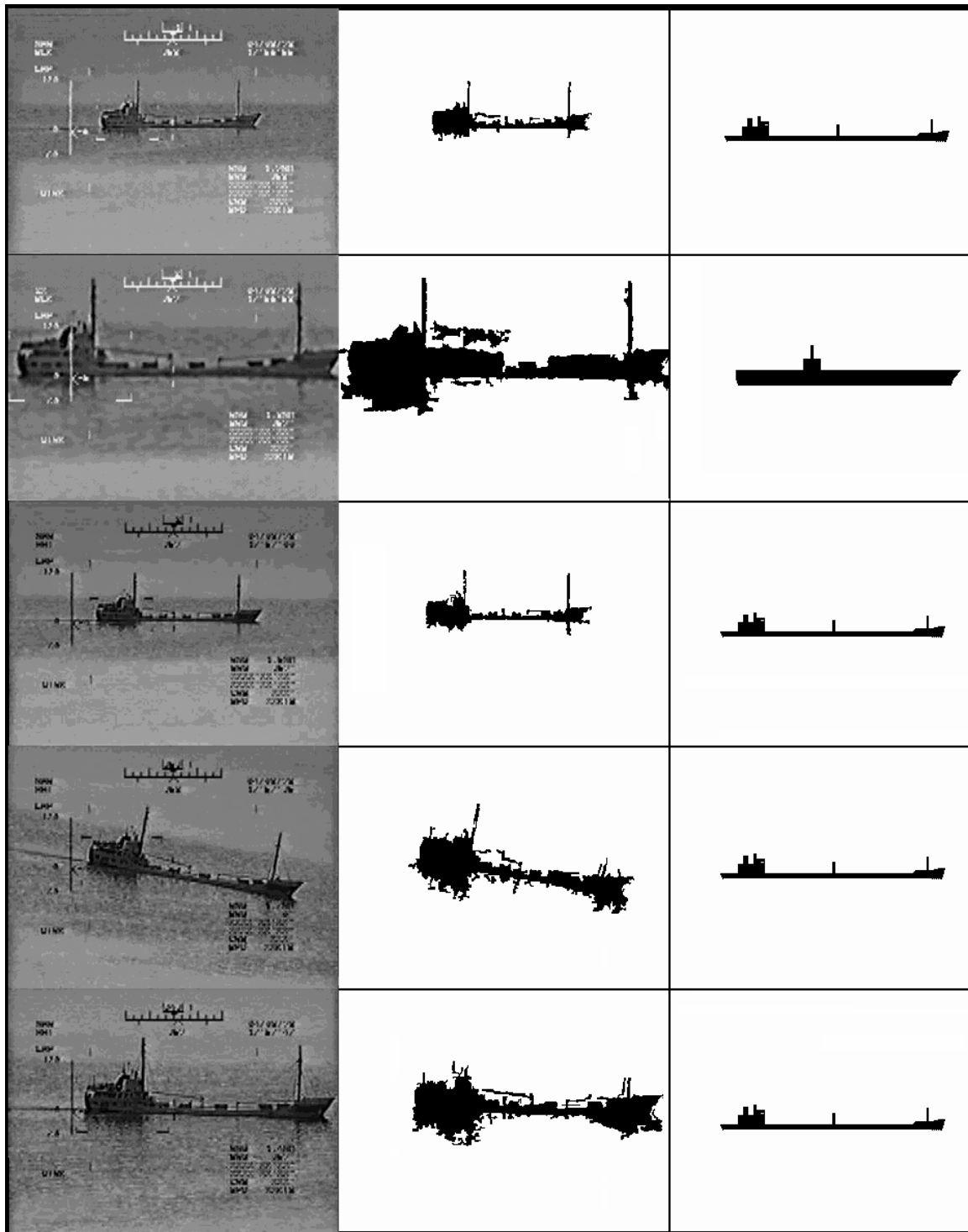
Figure 40: Classification results for merchant FLIR images

51
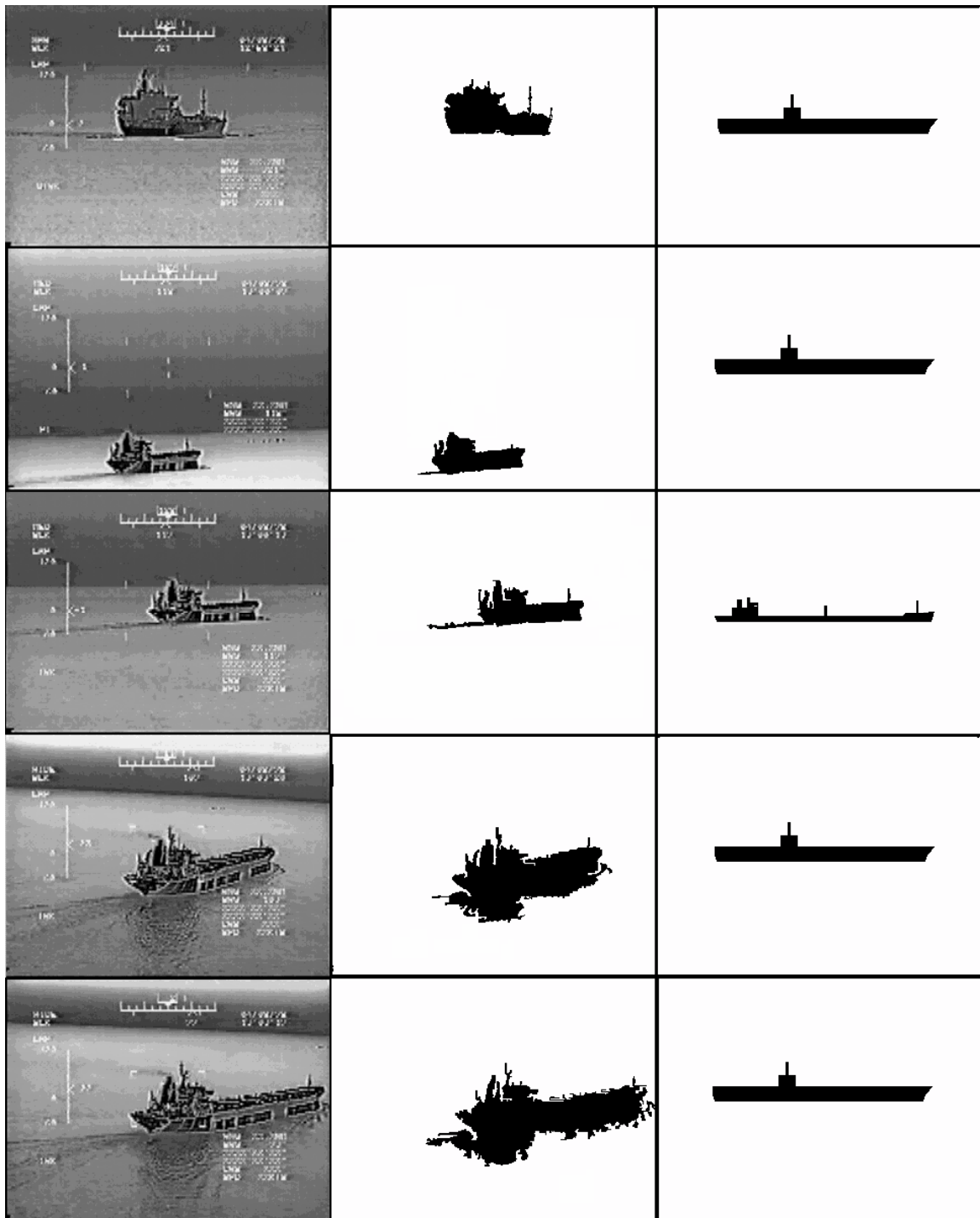
Figure 41: Classification results for merchant FLIR images
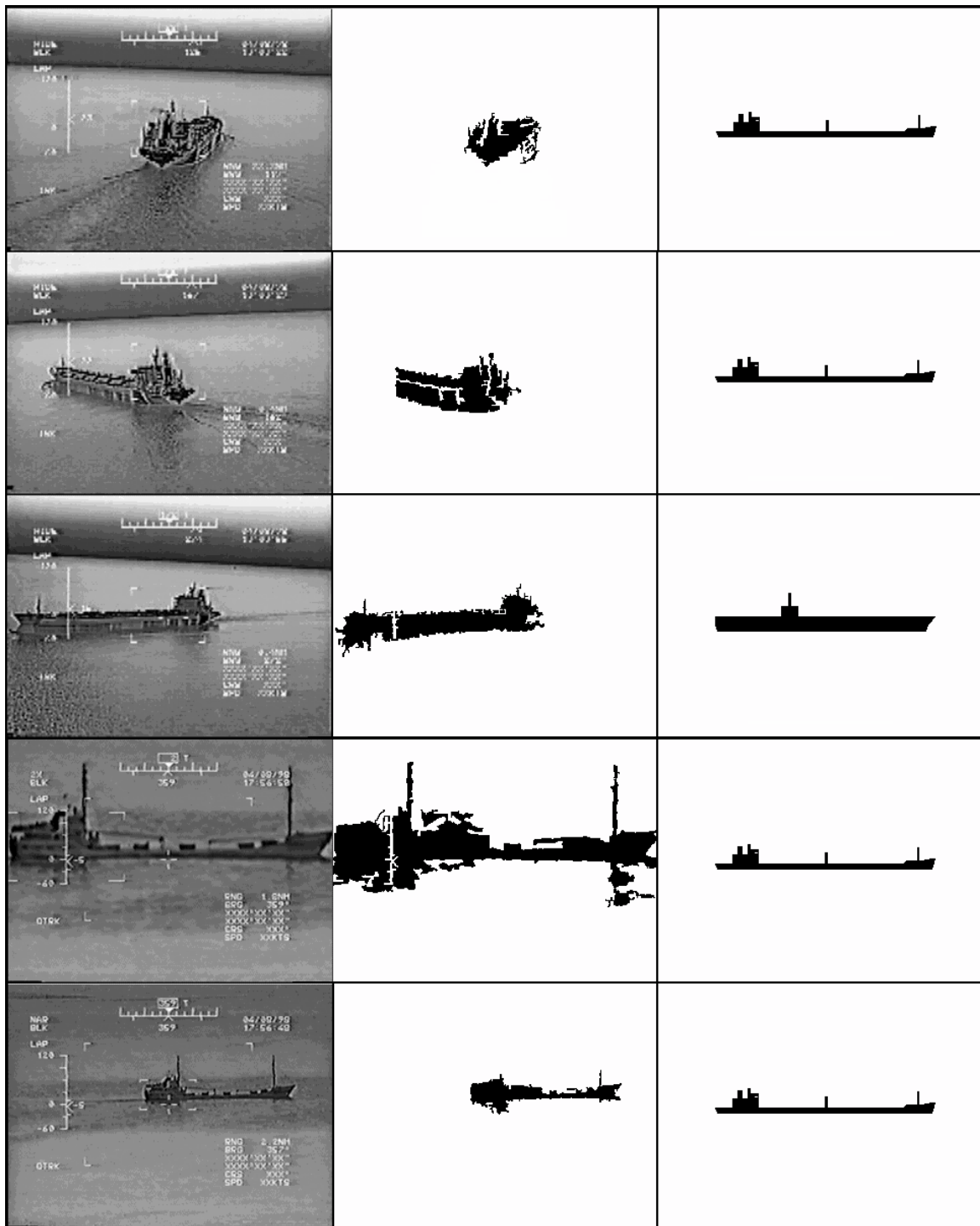
Figure 42: Classification results for merchant FLIR images
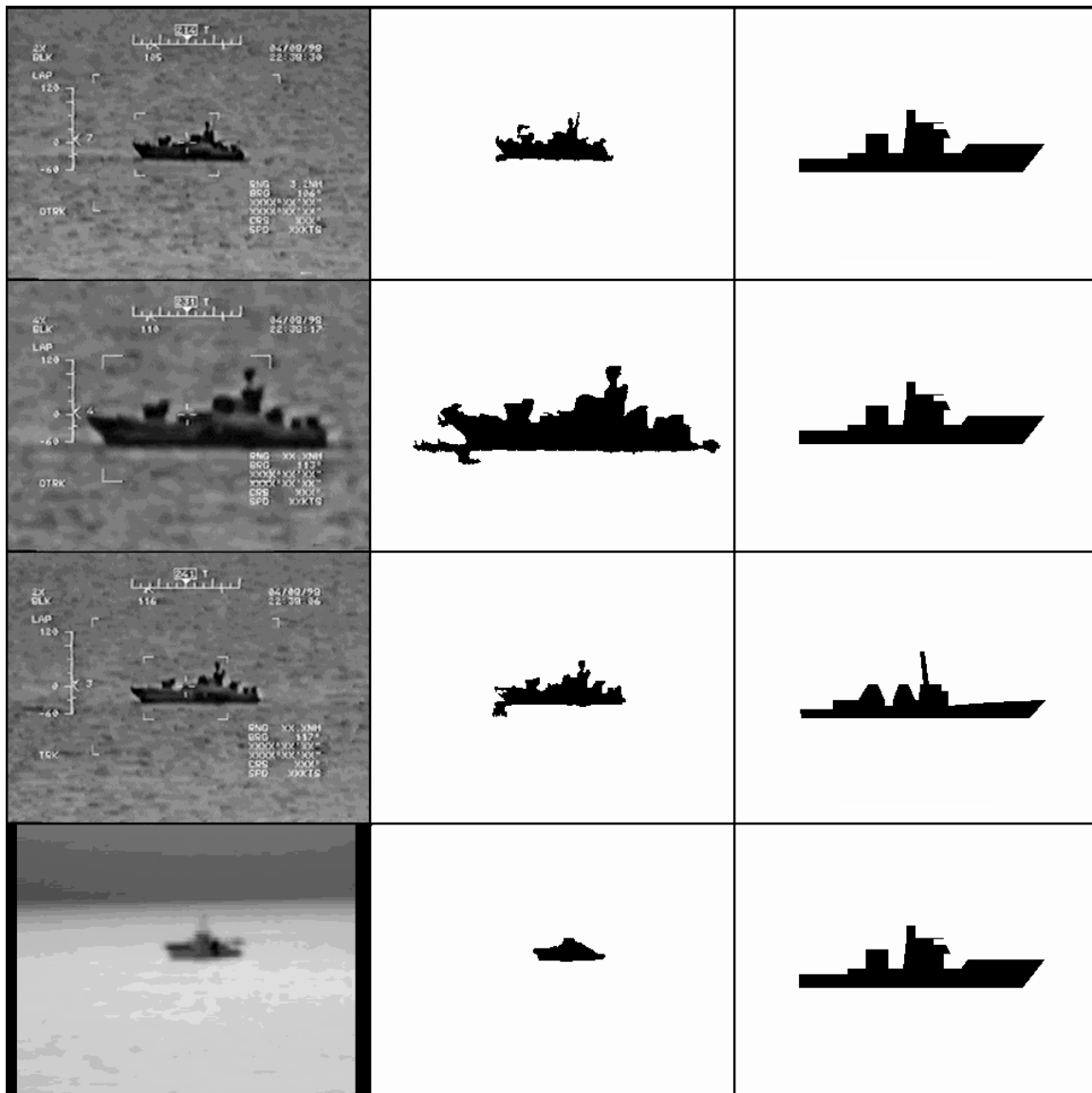
Figure 43: Classification results for point sur FLIR images
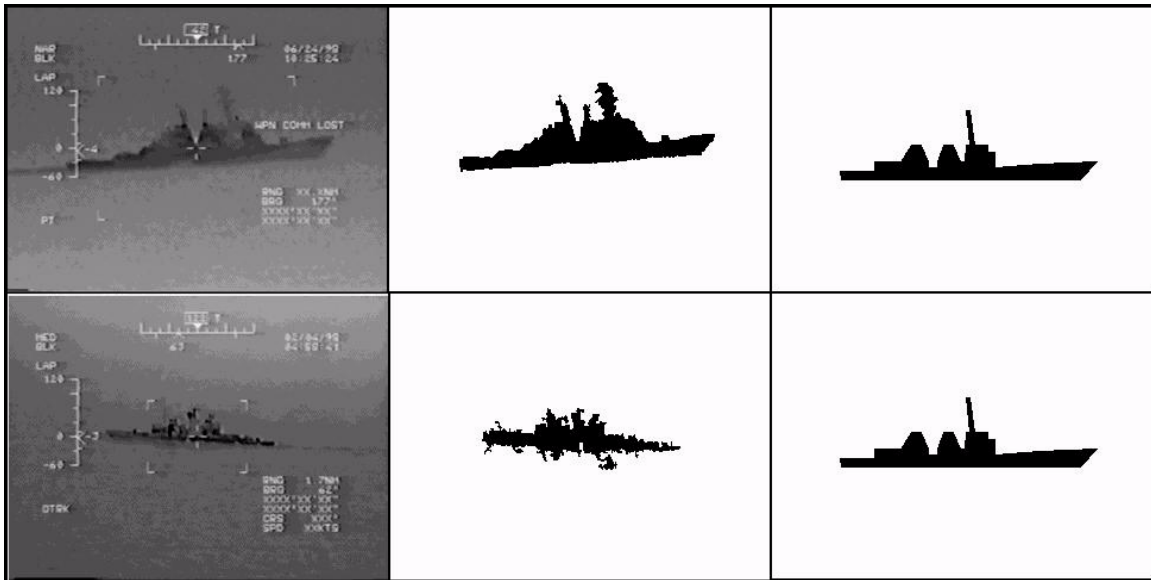
Figure 44: Classification results for destroyer FLIR

THIS PAGE INTENTIONALLY LEFT BLANK

VI.   CONCLUSIONS

This thesis explored a moment-based method for ship-type recognition. Numerical simulations were carried out with a set of five three-dimensional ship models. Moment-invariant signatures were used as the input to a neural-network classifier. The classifier achieved a 87.5% correct classification rate (within the set of test-models) for a complete range of point of view around the input ship model. The success is due to a combination of a robust feature extraction and the neural-network generalization capability.

A test with 25 real FLIR ship images was also done. Experimental results were worse due to the noisy extracted silhouettes. The maximum classification accuracy of 68% should only be considered a rough approximation to the sort of accuracies one can expect from a fully operational classifier.

A larger database of real FLIR ship images needs to be tested. The acquisition process should eliminate undesired alphanumeric data superimposed on the FLIR images. Using this database, our system could be trained using the real FLIR images from different viewpoints, and this could provide better performance on new real images. Another recommendation is to investigate different segmentation algorithms capable of addressing some predictable distortions like the ship shadow/reflection on the sea surface and the smoke coming out from stacks. As we can see in Figure 35, the merchant FLIR images presented the worst accuracy due to these distortions.

In conclusion, this thesis demonstrated how ship recognition using models is complicated by the imaging process, which involves a viewpoint-dependent two-dimensional projection of three-dimensional ship model. As a consequence, the appearance of a ship in an image can vary greatly with its aspect and scale. Ship recognition from infrared images is further complicated because the extracted ship silhouettes can be noisy due to the distortions caused by shadows, smoke, and other factors.

This thesis demonstrated the potential of a simple algorithm for this particular application. The modest requirements in terms of computer and FLIR hardware of this system show great potential for providing a recognition system to a variety of users.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. PROGRAM LISTING

```
%=============================================================================
%          Naval Postgraduate School - CA
%
%  Type   : Function
%  Name   : aircarrier.m
%  Function  : returns the vertices and faces to be used by the MATLAB function
%             "patch" and construct the aircraft carrier 3-D model
%  Date   01 march 2001
%  Version   : 1.0
%  Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%=============================================================================

function [verts,faces]=aircarrier();

verts=[122, -7,   0 %1
       122,   7,   0 %2
       116,   1,  -7 %3
       114,   0,  -12   %4
       116,  -1,  -7 %5
        77, -10,   0 %6
        77, -10,  -7 %7
        77,  -8, -12  %8
        68, -10,  -7  %9
        68, -21,   0 %10
       -23, -21,   0 %11
       -23, -10,  -7 %12
       -62, -10,   0 %13
       -62, -10,  -7 %14
       -60,  -8, -12   %15
       -62,  10,   0 %16
       -62,  10,  -7 %17
       -60,   8, -12   %18
       -36,  12,   0 %19
       -36,  10,  -7 %20
       -28,  21,   0 %21
       -28,  10,  -7 %22
        38,  21,   0 %23
        38,  10,  -7    %24
        65,  24,   0    %25
        65,  10,  -7 %26
        77,  10,   0 %27
        77,  10,  -7 %28
        77,   8, -12 %29
        -7, -19,   0 %30
        -7, -19,   9 %31
        -7, -15,   9 %32
        -7, -15,   0 %33
         7, -19,   0 %34
         7, -19,   9 %35
         7, -12,   9 %36
         7, -12,   0 %37
        -1, -17,   9 %38
        -1, -17,  20 %39
        -1, -15,  20 %40
        -1, -15,   9 %41
         1, -17,   9 %42
```

59

```
    1,    -17,   20 %43
    1,    -15,    9 %44
    1,    -15,   20];%45

%number =      32 12

plane1 =   [1,2,3,4,5,5,5,5,5,5,5,5];
plane2 =   [1,5,6,6,6,6,6,6,6,6,6,6];
plane3 =   [5,6,7,7,7,7,7,7,7,7,7,7];
plane4 =   [4,5,7,8,8,8,8,8,8,8,8,8];
plane5 =   [6,9,7,7,7,7,7,7,7,7,7,7];
plane6 =   [6,10,9,9,9,9,9,9,9,9,9,9];
plane7 =   [9,10,11,12,12,12,12,12,12,12,12,12];
plane8 =   [11,12,13,13,13,13,13,13,13,13,13,13];
plane9 =   [12,13,14,14,14,14,14,14,14,14,14,14];
plane10   =   [7,9,12,14,15,8,8,8,8,8,8,8];
plane11   =   [13,16,17,14,14,14,14,14,14,14,14,14];
plane12   =   [14,17,18,15,15,15,15,15,15,15,15,15];
plane13   =   [16,19,20,17,17,17,17,17,17,17,17,17];
plane14   =   [19,21,22,20,20,20,20,20,20,20,20,20];
plane15   =   [21,23,24,22,22,22,22,22,22,22,22,22];
plane16   =   [23,25,26,24,24,24,24,24,24,24,24,24];
plane17   =   [25,27,26,26,26,26,26,26,26,26,26,26];
plane18   =   [26,27,28,28,28,28,28,28,28,28,28,28];
plane19   =   [27,3,28,28,28,28,28,28,28,28,28,28];
plane20   =   [27,2,3,3,3,3,3,3,3,3,3,3];
plane21   =   [28,3,4,29,29,29,29,29,29,29,29,29];
plane22   =   [17,20,22,24,26,28,29,18,18,18,18,18];
flightdeck   =   [1,6,10,11,13,16,19,21,23,25,27,2];
bottom =   [4,8,15,18,29,29,29,29,29,29,29,29];
tower1 =   [30,31,32,33,33,33,33,33,33,33,33,33];
tower2 =   [30,31,35,34,34,34,34,34,34,34,34,34];
tower3   =   [34,35,36,37,37,37,37,37,37,37,37,37];
tower4 =   [32,33,37,36,36,36,36,36,36,36,36,36];
tower5 =   [31,32,36,35,35,35,35,35,35,35,35,35];
mast1   =   [38,39,40,41,41,41,41,41,41,41,41,41];
mast2   =   [38,39,43,42,42,42,42,42,42,42,42,42];
mast3   =   [42,44,45,43,43,43,43,43,43,43,43,43];
mast4   =   [41,44,45,40,40,40,40,40,40,40,40,40];
mast5   =   [39,43,45,40,40,40,40,40,40,40,40,40];
%faces

faces=[   plane1
plane2
plane3
plane4
plane5
plane6
plane7
plane8
plane9
plane10
plane11
plane12
plane13
plane14
plane15
plane16
plane17
plane18
```

60

```
    plane19
    plane20
    plane21
    plane22
    flightdeck
    bottom
    tower1
    tower2
    tower3
    tower4
    tower5
    mast1
    mast2
    mast3
    mast4
    mast5 ];
%=============================================================================
% End of file aircarrier.m
```

```matlab
%==========================================================================
%          Naval Postgraduate School - CA
%
%  Type   : Function
%  Name   : destroyer.m
%  Function  : returns the vertices and faces to be used by the MATLAB function
%             "patch" in order to construct the destroyer 3-D model
%  Date   01 march 2001
%  Version   : 1.0
%  Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%==========================================================================

function [verts,faces]=destroyer();

verts=[140, 0,    5  %1
       90,   -16, 3 %2
       90,   -16,  -10 %3
       125,   0,  -10  %4
       50,  -20,  0 %5
       50,  -20,   -10 %6
       44,  -20,  0 %7
       44,   -20, 5  %8
       38,  -20, 5   %9
       29,  -20,   5 %10
       8, -20,   5    %11
       8, -20,  0 %12
       -8, -20,   0  %13
       -8, -20,  -10 %14
       -8, -20, 5    %15
       -55,  -14,   5   %16
       -55,  -14,  -3   %17
       -55,   -13, -10   %18
       -85,  -10,   -3  %19
       -84,  -9,  -10   %20
       -85,  10,   -3   %21
       -84,  9,  -10 %22
       -55,   14,   -3  %23
       -55,   13,  -10  %24
       -55,   14,   5   %25
       -8,   20,   5 %26
       -8,   20,    0 %27
       -8,   20,  -10   %28
       8,    20, 0   %29
       8,  20,    5    %30
       29,  20,   5    %31
       38,  20,   5    %32
       44,  20,   5    %33
       44,   20,    0 %34
       50,   20,    0 %35
       50,   20,   -10  %36
       90,   16,    3 %37
       90,   15,   -10   %38
       50,  -11,  0 %39
       50,  11,  0   %40
       50,  -6,   5 %41
       44,   -6,   5 %42
       44,   -11,  5 %43
       38,   -20,   5    %44
       44,  -20,  5%45
       50,   -11,  5%46
```

62

```
    50,       6,   5%47
    44,       6,   5%48
    44,      11,   5%49
    38,      20,   5%50
    44,      20,   5%51
    50,      11,   5%52
    50,      -6,  14%53
    44,      -6,  14%54
    44,       6,  14%55
    50,       6,  14%56
    38,     -19,  20%57
    44,     -11,  20%58
    44,      11,  20%59
    38,      19,  20%60
    29,      19,  20%61
    25,      11,  20%62
    25,     -11,  20%63
    29,     -19,  20%64
     0,     -10,   5%65
    20,     -10,   5%66
    20,      10,   5%67
     0,      10,   5%68
    24,      11,   5%69
    24,     -11,   5%70
     6,      -4,  20%71
    14,      -4,  20%72
    14,       4,  20%73
     6,       4,  20%74
     0,     -10,   0%75
     0,      10,   0%76
   -32,     -10,   5%77
    -8,     -10,   5%78
    -8,      10,   5%79
   -32,      10,   5%80
   -22,      -4,  20%81
   -14,      -4,  20%82
   -14,       4,  20%83
   -22,       4,  20%84
    29,      -2,  20%85
    33,      -2,  20%86
    33,       2,  20%87
    29,       2,  20%88
    25,      -2,  50%89
    29,      -2,  50%90
    29,       2,  50%91
    25,       2,  50];%92


%number =     32 12

plane1 =   [1,2,3,4,4,4,4,4,4,4,4];
plane2 =   [2,5,6,3,3,3,3,3,3,3,3];
plane3 =   [5,7,8,9,10,11,12,13,14,6,6];
plane4 =   [13,15,16,17,18,14,14,14,14,14,14];
plane5 =   [17,19,20,18,18,18,18,18,18,18,18];
plane6 =   [19,21,22,20,20,20,20,20,20,20,20];
plane7 =   [21,23,24,22,22,22,22,22,22,22,22];
plane8 =   [23,25,26,27,28,24,24,24,24,24,24];
plane9 =   [27,29,30,31,32,33,34,35,36,28,28];
```

63

```
plane10    =   [35,37,38,36,36,36,36,36,36,36,36];
plane11    =   [37,1,4,38,38,38,38,38,38,38,38];
botton =   [4,3,6,14,18,20,22,24,28,36,38];
deck1  =   [1,2,37,37,37,37,37,37,37,37,37];
deck2  =   [2,5,35,37,37,37,37,37,37,37,37];
deck3  =   [5,7,39,39,39,39,39,39,39,39,39];
deck4  =   [34,35,40,40,40,40,40,40,40,40,40];
deck5  =   [46,41,42,43,9,8,8,8,8,8,8];
deck6  =   [47,48,49,32,51,52,52,52,52,52,52];
deck7  =   [53,54,55,56,56,56,56,56,56,56,56];
deck8  =   [57,58,59,60,61,62,63,64,64,64,64];
deck9  =   [10,11,65,66,67,68,30,31,69,70,70];
deck10     =   [13,12,75,76,29,27,27,27,27,27,27];
deck11     =   [16,15,78,77,80,79,26,25,25,25,25];
deck12     =   [19,17,23,21,21,21,21,21,21,21,21];
lateral1   =   [8,46,39,7,7,7,7,7,7,7,7];
lateral2   =   [39,46,41,53,56,47,52,40,40,40,40];
lateral3   =   [40,52,33,34,34,34,34,34,34,34,34];
lateral4   =   [54,53,41,42,42,42,42,42,42,42,42];
lateral5   =   [55,56,47,48,48,48,48,48,48,48,48];
lateral6   =   [58,59,49,48,55,54,42,43,43,43,43];
lateral7   =   [57,58,43,9,9,9,9,9,9,9,9];
lateral8   =   [60,59,49,32,32,32,32,32,32,32,32];
lateral9   =   [64,57,9,10,10,10,10,10,10,10,10];
lateral10=[61,60,32,31,31,31,31,31,31,31,31];
lateral11=[63,64,10,70,70,70,70,70,70,70,70];
lateral12=   [62,61,31,69,69,69,69,69,69,69,69];
lateral13=   [62,63,70,69,69,69,69,69,69,69,69];
lateral14=   [65,11,12,75,75,75,75,75,75,75,75];
lateral15=   [68,30,29,76,76,76,76,76,76,76,76];
lateral16=   [65,75,76,68,68,68,68,68,68,68,68];
lateral17=   [15,13,27,26,26,26,26,26,26,26,26];
lateral18=[16,17,23,25,25,25,25,25,25,25,25];
stack1     =   [71,72,66,65,65,65,65,65,65,65,65];
stack2     =   [74,73,67,68,68,68,68,68,68,68,68];
stack3     =   [65,71,74,68,68,68,68,68,68,68,68];
stack4     =   [66,72,73,67,67,67,67,67,67,67,67];
stack5     =   [71,72,73,74,74,74,74,74,74,74,74];
stack6     =   [77,81,82,78,78,78,78,78,78,78,78];
stack7     =   [80,84,83,79,79,79,79,79,79,79,79];
stack8     =   [77,81,84,80,80,80,80,80,80,80,80];
stack9     =   [78,82,83,79,79,79,79,79,79,79,79];
stack10    =   [81,82,83,84,84,84,84,84,84,84,84];
mast1      =   [85,86,90,89,89,89,89,89,89,89,89];
mast2      =   [86,87,91,90,90,90,90,90,90,90,90];
mast3      =   [87,88,92,91,91,91,91,91,91,91,91];
mast4      =   [88,85,89,92,92,92,92,92,92,92,92];
mast5      =   [89,90,91,92,92,92,92,92,92,92,92];

%faces

faces=[plane1
   plane2
   plane3
   plane4
   plane5
   plane6
   plane7
   plane8
   plane9
```

```
            plane10
            plane11
            botton
            deck1
            deck2
            deck3
            deck4
            deck5
            deck6
            deck7
            deck8
            deck9
            deck10
            deck11
            deck12
            lateral1
            lateral2
            lateral3
            lateral4
            lateral5
            lateral6
            lateral7
            lateral8
            lateral9
            lateral10
            lateral11
            lateral12
            lateral13
            lateral14
            lateral15
            lateral16
            lateral17
            lateral18
            stack1
            stack2
            stack3
            stack4
            stack5
            stack6
            stack7
            stack8
            stack9
            stack10
            mast1
            mast2
            mast3
            mast4
            mast5];
%========================================================================
% End of file destroyer.m
```

```matlab
%===========================================================================
%           Naval Postgraduate School - CA
%
%   Type   : Function
%   Name   : frigate.m
%   Function  : returns the vertices and faces to be used by the MATLAB function
%               "patch" in order to construct the frigate 3-D model
%   Date   01 march 2001
%   Version   : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%===========================================================================

function [verts,faces]=frigate();

verts=[128, 0,   16 %1
        96,   -13, 14 %2
        96,   -7,  0 %3
        112,   0,  0 %4
        92,  -15,  12 %5
        92,  -9,   0 %6
        60,  -20,  10 %7
        60,   -18, 0  %8
        58,  -20,  10  %9
        48,  -20,  10 %10
       -26, -20,  10 %11
       -60, -20,  10 %12
       -60, -18,   0 %13
       -102,-15,  10 %14
        -98,  -13,  0    %15
       -102,  15, 10 %16
        -98,  13,  0 %17
       -60,   20, 10    %18
       -60,  18,   0 %19
       -26,  20,  10 %20
        48,  20,   10 %21
        58,  20,   10 %22
        60,  20,   10     %23
        60,18,0     %24
        92,15,12    %25
        92,9,0 %26
        96,13,14    %27
        96,7,0 %28
        60,-13,10 %29
        60,13,10   %30
       -12,-12,10 %31
        8,-12,10   %32
        8,-16,10   %33
       -12,12,10 %34
        8,12,10    %35
        8,16,10    %36
        58,-20,20 %37
        48,-20,20 %38
        48,-16,20 %39
        8,-16,20   %40
        8,-12,20   %41
       -12,-12,20 %42
       -26,-20,20 %43
       -60,-20,20 %44
       -60,20,20    %45
       -26,20,20    %46
```

66

```
-12,12,20    %47
8,12,20      %48
8,16,20      %49
48,16,20     %50
48,20,20     %51
58,20,20     %52
60,13,20     %53
60,-13,20    %54
57,-13,25    %55
43,-13,25    %56
43,13,25     %57
57,13,25     %58
60,7,25      %59
60,-7,25     %60
49,-20,20    %61
49,-20,12    %62
51,-20,12    %63
51,-20,20    %64
52,-20,20    %65
52,-20,12    %66
56,-20,22    %67
58,-20,22    %68
49,20,20     %69
49,20,12     %70
51,20,12     %71
51,20,20     %72
52,20,20     %73
52,20,12     %74
56,20,22     %75
58,20,22     %76
60,-13,22    %77
60,-7,22     %78
60,7,22      %79
60,13,22     %80
60,-7,20     %81
57,-13,20    %82
60,7,20      %83
57,13,20     %84
43,-13,20    %85
43,13,20     %86
59,-16,20    %87
59,-16,10    %88
59,16,20     %89
59,16,10     %90
-39,-5,20    %91
-37,-4,25    %92
-37,4,25     %93
-39,5,20     %94
-29,-4,25    %95
-28,-5,20    %96
-29,4,25     %97
-28,5,20     %98
-16,-3,24    %99
-10,-3,24     %100
-10,-3,20     %101
-16,-3,20    %102
-16,3,24     %103
-10,3,24      %104
-10,3,20      %105
-16,3,20     %106
```

```
        18,-1,48     %107
        20,-1,48     %108
        20,1,48      %109
        18,1,48      %110
        18,-1,20     %111
        20,-1,20     %112
        20,1,20      %113
        18,1,20      %114
        36,-2,42     %115
        39,-2,42     %116
        39,2,42      %117
        36,2,42      %118
        36,-2,20     %119
        39,-2,20     %120
        39,2,20      %121
        36,2,20      %122
        48,-2,33     %123
        52,-2,33     %124
        52,2,33      %125
        48,2,33      %126
        48,-2,25     %127
        52,-2,25     %128
        52,2,25      %129
        48,2,25];       %130
%number =    66 planes
plane1 =   [1,2,3,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4];
plane2 =   [2,5,6,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3];
plane3 =   [5,7,8,6,6,6,6,6,6,6,6,6,6,6,6,6,6,6];
plane4 =   [7,9,10,11,12,13,8,8,8,8,8,8,8,8,8,8,8,8];
plane5 =   [12,14,15,13,13,13,13,13,13,13,13,13,13,13,13,13,13,13];
plane6 =   [14,16,17,15,15,15,15,15,15,15,15,15,15,15,15,15,15,15];
plane7 =   [16,18,19,17,17,17,17,17,17,17,17,17,17,17,17,17,17,17];
plane8 =   [18,20,21,22,23,24,19,19,19,19,19,19,19,19,19,19,19,19];
plane9 =   [23,25,26,24,24,24,24,24,24,24,24,24,24,24,24,24,24,24];
plane10   = [25,27,28,26,26,26,26,26,26,26,26,26,26,26,26,26,26,26];
plane11   = [27,1,4,28,28,28,28,28,28,28,28,28,28,28,28,28,28,28];
botton =   [4,3,6,8,13,15,17,19,24,26,28,28,28,28,28,28,28,28];
deck1  =   [1,2,27,27,27,27,27,27,27,27,27,27,27,27,27,27,27,27];
deck2  =   [2,5,25,27,27,27,27,27,27,27,27,27,27,27,27,27,27,27];
deck3  =   [5,7,9,29,30,22,23,25,25,25,25,25,25,25,25,25,25,25];
deck4  =   [9,10,11,31,32,33,88,88,88,88,88,88,88,88,88,88,88,88];
deck5  =   [22,21,20,34,35,36,90,90,90,90,90,90,90,90,90,90,90,90];
deck6  =   [37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54];
deck7  =   [12,14,16,18,18,18,18,18,18,18,18,18,18,18,18,18,18,18];
deck8  =   [55,56,57,58,59,60,60,60,60,60,60,60,60,60,60,60,60,60];
lateral1   = [9,10,38,61,62,63,64,65,66,67,68,68,68,68,68,68,68,68];
lateral2   = [22,21,51,69,70,71,72,73,74,75,76,76,76,76,76,76,76,76];
lateral3   = [77,78,60,59,79,80,30,29,29,29,29,29,29,29,29,29,29,29];
lateral4   = [68,77,29,9,9,9,9,9,9,9,9,9,9,9,9,9,9,9];
lateral5   = [76,80,30,22,22,22,22,22,22,22,22,22,22,22,22,22,22,22];
lateral6   = [55,60,81,82,82,82,82,82,82,82,82,82,82,82,82,82,82,82];
lateral7   = [58,59,83,84,84,84,84,84,84,84,84,84,84,84,84,84,84,84];
lateral8   = [56,55,82,85,85,85,85,85,85,85,85,85,85,85,85,85,85,85];
lateral9   = [57,58,84,86,86,86,86,86,86,86,86,86,86,86,86,86,86,86];
lateral10=[56,57,86,85,85,85,85,85,85,85,85,85,85,85,85,85,85,85];
lateral11=[87,88,33,40,40,40,40,40,40,40,40,40,40,40,40,40,40,40];
lateral12=   [89,90,36,49,49,49,49,49,49,49,49,49,49,49,49,49,49,49];
lateral13=   [40,33,32,41,41,41,41,41,41,41,41,41,41,41,41,41,41,41];
lateral14=   [49,36,35,48,48,48,48,48,48,48,48,48,48,48,48,48,48,48];
lateral15=   [31,42,41,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32];
```

```
   lateral16=    [34,47,48,35,35,35,35,35,35,35,35,35,35,35,35,35,35,35];
    lateral17=    [43,11,31,42,42,42,42,42,42,42,42,42,42,42,42,42,42,42];
   lateral18=[46,20,34,47,47,47,47,47,47,47,47,47,47,47,47,47,47,47];
   lateral19=[44,43,11,12,12,12,12,12,12,12,12,12,12,12,12,12,12,12];
   lateral20=[45,46,20,18,18,18,18,18,18,18,18,18,18,18,18,18,18,18];
   lateral21=[44,12,18,45,45,45,45,45,45,45,45,45,45,45,45,45,45,45];
   stack1   = [91,92,93,94,94,94,94,94,94,94,94,94,94,94,94,94,94,94];
   stack2   = [91,92,95,96,96,96,96,96,96,96,96,96,96,96,96,96,96,96];
   stack3   = [96,95,97,98,98,98,98,98,98,98,98,98,98,98,98,98,98,98];
   stack4   = [94,93,97,98,98,98,98,98,98,98,98,98,98,98,98,98,98,98];
   stack5=[92,95,97,93,93,93,93,93,93,93,93,93,93,93,93,93,93,93];
turret1=[99,100,101,102,102,102,102,102,102,102,102,102,102,102,102,102,102,102
];
turret2=[100,104,105,101,101,101,101,101,101,101,101,101,101,101,101,101,101,10
1];
turret3=[103,104,105,106,106,106,106,106,106,106,106,106,106,106,106,106,106,10
6];
turret4=[99,102,106,103,103,103,103,103,103,103,103,103,103,103,103,103,103,103
];
turret5=[99,100,104,103,103,103,103,103,103,103,103,103,103,103,103,103,103,103
];
mast1=[107,108,112,111,111,111,111,111,111,111,111,111,111,111,111,111,111,111]
;
mast2=[108,109,113,112,112,112,112,112,112,112,112,112,112,112,112,112,112,112]
;
mast3=[109,110,114,113,113,113,113,113,113,113,113,113,113,113,113,113,113,113]
;
mast4=[110,107,111,114,114,114,114,114,114,114,114,114,114,114,114,114,114,114]
;
mast5=[107,108,109,110,110,110,110,110,110,110,110,110,110,110,110,110,110,110]
;
surface1=[115,116,120,119,119,119,119,119,119,119,119,119,119,119,119,119,119,1
19];
surface2=[116,117,121,120,120,120,120,120,120,120,120,120,120,120,120,120,120,1
20];
surface3=[117,118,122,121,121,121,121,121,121,121,121,121,121,121,121,121,121,1
21];
surface4=[118,115,119,122,122,122,122,122,122,122,122,122,122,122,122,122,122,1
22];
surface5=[115,116,117,118,118,118,118,118,118,118,118,118,118,118,118,118,118,1
18];
radar1=[123,124,128,127,127,127,127,127,127,127,127,127,127,127,127,127,127,127
];
radar2=[124,125,129,128,128,128,128,128,128,128,128,128,128,128,128,128,128,128
];
radar3=[125,126,130,129,129,129,129,129,129,129,129,129,129,129,129,129,129,129
];
radar4=[126,123,127,130,130,130,130,130,130,130,130,130,130,130,130,130,130,130
];
radar5=[123,124,125,126,126,126,126,126,126,126,126,126,126,126,126,126,126,126
];
   %faces  66 planes
   faces=[plane1
      plane2
      plane3
      plane4
      plane5
      plane6
      plane7
      plane8
```

```
        plane9
        plane10
        plane11
        botton
        deck1
        deck2
        deck3
        deck4
        deck5
        deck6
        deck7
        deck8
        lateral1
        lateral2
        lateral3
        lateral4
        lateral5
        lateral6
        lateral7
        lateral8
        lateral9
        lateral10
        lateral11
        lateral12
        lateral13
        lateral14
        lateral15
        lateral16
        lateral17
        lateral18
        lateral19
        lateral20
        lateral21
        stack1
        stack2
        stack3
        stack4
        stack5
        turret1
        turret2
        turret3
        turret4
        turret5
        mast1
        mast2
        mast3
        mast4
        mast5
        surface1
        surface2
        surface3
        surface4
        surface5
        radar1
        radar2
        radar3
        radar4
        radar5];
%========================================================================
% End of file frigate.m
```

```matlab
%==============================================================================
%          Naval Postgraduate School - CA
%
%   Type   : Function
%   Name   : pointsur.m
%   Function  : returns the vertices and faces to be used by the MATLAB function
%               "patch" in order to construct the research ship (point sur) 3-D
model
%   Date   01 march 2001
%   Version   : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%==============================================================================

function [verts,faces]=pointsur();


verts=[93,    0,0   %0
       81, 0, -15    %1
       39,16, -15    %2
       -43,   16, -15   %3
       -43,-16, -15 %4
       39, -16, -15 %5
       51, -16,  0  %6
       47, -16,  -5 %7
       37, -10,  -5  %8
        37,-16,  12  %9
       17, -16,  12 %10
       15, -16,  -5 %11
       -43,-16,  -8 %12
       -43,  16,  -8   %13
       15, 16,  -5   %14
       17, 16,  12   %15
       37, 16,  12   %16
       37, 10,  -5   %17
       47, 16,  -5   %18
       51, 16,0  %19
       21,   3,  12 %20
       17, 3,  12   %21
       17, -3,  12   %22
       21, -3,  12   %23
       21, 3,  19    %24
       17, 3,  19    %25
       17, -3,  19   %26
       21, -3,  19   %27
       6,  16,  -8   %28
       -6, 16,  -8   %29
       -6, 8,  -8   %30
       6,  8,  -8   %31
       6,  16,  6   %32
       -6, 16,6  %33
       -6, 8,6  %34
       6,  8,6  %35
       -16,   16,  -8   %36
       -16,   16,  -5   %37
       -16,-16,  -5 %38
       -16,-16,  -8 %39
       38, -16,  8   %40
       38, 16,8  %41
       31, 16,8  %42
       31, 16,  12   %43
```

71

```
       31, -16,  12  %44
        31, -16,   8  %45
        17, -16,  -8  %46
        17, 16,  -8  %47
        41, -16,   4  %48
        41, 16, 4  %49
        33, 16, 4  %50
        33, -16,   4   %51
        31, -16,  -5 %52
        31, 16,  -5   %53
        31, 16, 4  %54
        27, 16, 4  %55
        25, 16,  -5   %56
        29, 16,  -5   %57
        25, 16, 4  %58
        21, 16, 4  %59
        19, 16,  -5   %60
        23, 16,  -5   %61
        31, -16,   4   %62
        27, -16,   4   %63
        25, -16,  -5 %64
        29, -16,  -5 %65
        25, -16,   4   %66
        21, -16,   4   %67
        19, -16,  -5 %68
        23, -16,  -5 %69
        37, 10, 4  %70
        37, -10,   4   %71
        17, -10,   4   %72
        17, -10,  -5 %73
        17, 10,  -5   %74
        17, 10, 4]; %75

%number =     32

plane0 =  [0,1,2,18,19,19,19,19,19,19,19,19,19,19,19,19];
plane1 =  [2,3,13,47,14,18,18,18,18,18,18,18,18,18,18,18];
plane2 =  [4,5,7,11,46,12,12,12,12,12,12,12,12,12,12,12];
plane3 =  [0,6,7,5,1,1,1,1,1,1,1,1,1,1,1,1];
late0  =  [47,14,37,36,36,36,36,36,36,36,36,36,36,36,36,36];
late1  =  [8,71,72,73,73,73,73,73,73,73,73,73,73,73,73,73];
late2  =  [17,70,75,74,74,74,74,74,74,74,74,74,74,74,74,74];
late3  =  [52,51,48,40,45,44,10,11,68,67,66,69,64,63,62,65];
late4  =  [53,50,49,41,42,43,15,14,60,59,58,61,56,55,54,53];
bottom =  [1,2,3,4,5,5,5,5,5,5,5,5,5,5,5,5];
glass0 =  [48,40,41,49,49,49,49,49,49,49,49,49,49,49,49,49];
glass1 =  [45,44,43,42,42,42,42,42,42,42,42,42,42,42,42,42];
glass2 =  [8,17,70,71,71,71,71,71,71,71,71,71,71,71,71,71];
radar0 =  [20,24,27,23,23,23,23,23,23,23,23,23,23,23,23,23];
radar1 =  [20,21,25,24,24,24,24,24,24,24,24,24,24,24,24,24];
radar2 =  [22,23,27,26,26,26,26,26,26,26,26,26,26,26,26,26];
stack0 =  [28,32,35,31,31,31,31,31,31,31,31,31,31,31,31,31];
stack1 =  [32,33,29,28,28,28,28,28,28,28,28,28,28,28,28,28];
stack2 =  [30,31,35,34,34,34,34,34,34,34,34,34,34,34,34,34];
stack3 =  [30,34,33,29,29,29,29,29,29,29,29,29,29,29,29,29];
deck0  =  [0,19,6,6,6,6,6,6,6,6,6,6,6,6,6,6];
deck1  =  [6,7,18,19,19,19,19,19,19,19,19,19,19,19,19,19];
deck2  =  [7,8,17,18,18,18,18,18,18,18,18,18,18,18,18,18];
deck3  =  [15,16,9,10,22,23,20,21,21,21,21,21,21,21,21,21];
deck4  =  [24,25,26,27,27,27,27,27,27,27,27,27,27,27,27,27];
```

```
    deck5  =   [14,15,21,25,26,22,10,11,11,11,11,11,11,11,11,11];
    deck6  =   [11,38,37,29,30,31,28,14,14,14,14,14,14,14,14,14];
    deck7  =   [36,37,38,39,39,39,39,39,39,39,39,39,39,39,39,39];
    deck8  =   [36,13,12,39,39,39,39,39,39,39,39,39,39,39,39,39];
    deck9  =   [3,4,12,13,13,13,13,13,13,13,13,13,13,13,13,13];
    deck10 =   [32,33,34,35,35,35,35,35,35,35,35,35,35,35,35,35];
    deck11 =   [40,45,42,41,41,41,41,41,41,41,41,41,41,41,41,41];

    %faces

    faces=[    plane0
        plane1
        plane2
        plane3
        late0
        late1
        late2
        late3
        late4
        bottom
        glass0
        glass1
        glass2
        radar0
        radar1
        radar2
        stack0
        stack1
        stack2
        stack3
        deck0
        deck1
        deck2
        deck3
        deck4
        deck5
        deck6
        deck7
        deck8
        deck9
        deck10
        deck11];

faces=faces+1;
%=========================================================================
% End of file pointsur.m
```

```matlab
%=============================================================================
%           Naval Postgraduate School - CA
%
%   Type   : Function
%   Name   : merchant.m
%   Function  : returns the vertices and faces to be used by the MATLAB function
%               "patch" in order to construct the merchant 3-D model
%   Date   01 march 2001
%   Version   : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%=============================================================================

function [verts,faces]=merchant();

verts=[115,-11,0      %1
    115,-15,10        %2
    118.9,-14.5,10.8  %3
    117.8,-10.6,0     %4
    122.5,-13,11.5    %5
    120.5,-9.5,0      %6
    125.6,-10.6,12    %7
    122.8,-7.8,0      %8
    128,-7.5,12.5     %9
    124.5,-5.5,0      %10
    129.5,-3.9,13     %11
    125.6,-2.85,0     %12
    130,0,13          %13
    126,0,0           %14
    129.5,3.9,13      %15
    125.6,2.85,0      %16
    128,7.5,12.5      %17
    124.5,5.5,0       %18
    125.6,10.6,12     %19
    122.8,7.8,0       %20
    122.5,13,11.5     %21
    120.5,9.5,0       %22
    118.9,14.5,10.8   %23
    117.8,10.6,0      %24
    115,15,10         %25
    115,11,0          %26
    96,-15,10         %27
    93,-15,7          %28
    -130,-15,7        %29
    -128,-11,0        %30
    -128,11,0         %31
    -130,15,7         %32
    93,15,7           %33
    96,15,10          %34
    -80,-15,23        %35
    -80,15,23         %36
    -84,15,23         %37
    -84,11,23         %38
    -91,11,23         %39
    -91,-11,23        %40
    -84,-11,23        %41
    -84,-15,23        %42
    -91,-11,17        %43
    -91,11,17         %44
    -110,11,17        %45
    -110,-11,17       %46
```

```
-110,11,7        %47
-110,-11,7        %48
-80,-15,21       %49
-80,-11,19       %50
-80,-11,7        %51
-80,11,7         %52
-80,11,19        %53
-80,15,21        %54
-84,-15,21       %55
-84,-11,19       %56
-91,-11,7        %57
-84,15,21        %58
-84,11,19        %59
-91,11,7         %60
-99,-3,17        %61
-99,3,17         %62
-104,3,17        %63
-104,-3,17       %64
-99,3,27         %65
-99,-3,27        %66
-104,3,27        %67
-104,-3,27       %68
2,-11,7          %69
2,-9,7           %70
0,-9,7           %71
0,-11,7          %72
2,-11,20         %73
2,-9,20          %74
0,-11,20         %75
0,-9,20          %76
2,9,7            %77
2,11,7           %78
0,11,7           %79
0,9,7            %80
2,9,20           %81
2,11,20          %82
0,11,20          %83
0,9,20           %84
111,-1,10        %85
111,1,10         %86
109,1,10         %87
109,-1,10        %88
111,-1,26        %89
111,1,26         %90
109,1,26         %91
109,-1,26        %92
-89,-1,23        %93
-89,1,23         %94
-91,1,23         %95
-91,-1,23        %96
-89,-1,30        %97
-89,1,30         %98
-91,1,30         %99
-91,-1,30];      %100

%planes

proa1 = [1,2,3,4,4,4,4,4,4,4,4,4,4,4,4];
proa2 = [3,5,6,4,4,4,4,4,4,4,4,4,4,4,4];
proa3 = [5,7,8,6,6,6,6,6,6,6,6,6,6,6,6];
```

```
proa4 = [7,9,10,8,8,8,8,8,8,8,8,8,8,8,8];
proa5 = [9,11,12,10,10,10,10,10,10,10,10,10,10,10,10];
proa6 = [11,13,14,12,12,12,12,12,12,12,12,12,12,12,12];
proa7 = [13,15,16,14,14,14,14,14,14,14,14,14,14,14,14];
proa8 = [15,17,18,16,16,16,16,16,16,16,16,16,16,16,16];
proa9 = [17,19,20,18,18,18,18,18,18,18,18,18,18,18,18];
proa10= [19,21,22,20,20,20,20,20,20,20,20,20,20,20,20];
proa11= [21,23,24,22,22,22,22,22,22,22,22,22,22,22,22];
proa12= [23,25,26,24,24,24,24,24,24,24,24,24,24,24,24];
plane1 = [1,2,27,28,29,30,30,30,30,30,30,30,30,30,30];
plane2 = [29,30,31,32,32,32,32,32,32,32,32,32,32,32,32];
plane3 = [26,25,34,33,32,31,31,31,31,31,31,31,31,31,31];
deck1 = [2,3,5,7,9,11,13,15,17,19,21,23,25,25,25];
deck2 = [2,25,34,27,27,27,27,27,27,27,27,27,27,27,27];
deck3 = [27,34,33,28,28,28,28,28,28,28,28,28,28,28,28];
deck4 = [28,33,32,29,29,29,29,29,29,29,29,29,29,29,29];
deck5 = [35,36,37,38,39,40,41,42,42,42,42,42,42,42,42];
deck6 = [40,39,44,43,43,43,43,43,43,43,43,43,43,43,43];
deck7 = [43,44,45,46,46,46,46,46,46,46,46,46,46,46,46];
deck8 = [46,45,47,48,48,48,48,48,48,48,48,48,48,48,48];
botton = [1,4,6,8,10,12,14,16,18,20,22,24,26,31,30];
front = [35,49,50,51,52,53,54,36,36,36,36,36,36,36,36];
lateral1 = [51,50,56,41,40,43,57,57,57,57,57,57,57,57,57];
lateral2 = [52,53,59,38,39,44,60,60,60,60,60,60,60,60,60];
lateral3 = [43,46,48,57,57,57,57,57,57,57,57,57,57,57,57];
lateral4 = [44,45,47,60,60,60,60,60,60,60,60,60,60,60,60];
blocoright1 = [35,49,55,42,42,42,42,42,42,42,42,42,42,42,42];
blocoright2 = [41,42,55,56,56,56,56,56,56,56,56,56,56,56,56];
blocoleft1 = [36,37,58,54,54,54,54,54,54,54,54,54,54,54,54];
blocoleft2 = [37,38,59,58,54,54,54,54,54,54,54,54,54,54,54];
stack1 = [64,61,66,68,68,68,68,68,68,68,68,68,68,68,68];
stack2 = [61,62,65,66,66,66,66,66,66,66,66,66,66,66,66];
stack3 = [62,65,67,63,63,63,63,63,63,63,63,63,63,63,63];
stack4 = [67,68,64,63,63,63,63,63,63,63,63,63,63,63,63];
stacktop = [66,65,67,68,68,68,68,68,68,68,68,68,68,68,68];
middleright1 = [72,69,73,75,75,75,75,75,75,75,75,75,75,75,75];
middleright2 = [69,70,74,73,73,73,73,73,73,73,73,73,73,73,73];
middleright3 = [70,74,76,71,71,71,71,71,71,71,71,71,71,71,71];
middleright4 = [71,76,75,72,72,72,72,72,72,72,72,72,72,72,72];
middlerighttop = [73,74,75,76,76,76,76,76,76,76,76,76,76,76,76];
middleleft1 = [80,77,81,84,84,84,84,84,84,84,84,84,84,84,84];
middleleft2 = [77,78,82,81,81,81,81,81,81,81,81,81,81,81,81];
middleleft3 = [78,79,83,82,82,82,82,82,82,82,82,82,82,82,82];
middleleft4 = [79,80,84,83,83,83,83,83,83,83,83,83,83,83,83];
middlelefttop = [81,82,83,84,84,84,84,84,84,84,84,84,84,84,84];
mastproa1 = [88,85,89,92,92,92,92,92,92,92,92,92,92,92,92];
mastproa2 = [85,86,90,89,89,89,89,89,89,89,89,89,89,89,89];
mastproa3 = [86,87,91,90,90,90,90,90,90,90,90,90,90,90,90];
mastproa4 = [87,88,92,91,91,91,91,91,91,91,91,91,91,91,91];
mastproatop = [89,90,91,92,92,92,92,92,92,92,92,92,92,92,92];
mastpopa1 = [96,93,97,100,100,100,100,100,100,100,100,100,100,100,100];
mastpopa2 = [93,94,98,97,97,97,97,97,97,97,97,97,97,97,97];
mastpopa3 = [94,98,99,95,95,95,95,95,95,95,95,95,95,95,95];
mastpopa4 = [95,99,100,96,96,96,96,96,96,96,96,96,96,96,96];
mastpopatop = [97,98,99,100,100,100,100,100,100,100,100,100,100,100,100];

%faces

faces=[proa1
    proa2
```

```
        proa3
        proa4
        proa5
        proa6
        proa7
        proa8
        proa9
        proa10
        proa11
        proa12
        plane1
        plane2
        plane3
        deck1
        deck2
        deck3
        deck4
        deck5
        deck6
        deck7
        deck8
        botton
        front
        lateral1
        lateral2
        lateral3
        lateral4
        blocoright1
        blocoright2
        blocoleft1
        blocoleft2
        stack1
        stack2
        stack3
        stack4
        stacktop
        middleright1
        middleright2
        middleright3
        middleright4
        middlerighttop
        middleleft1
        middleleft2
        middleleft3
        middleleft4
        middlelefttop
        mastproa1
        mastproa2
        mastproa3
        mastproa4
        mastproatop
        mastpopa1
        mastpopa2
        mastpopa3
        mastpopa4
        mastpopatop];
%=========================================================================
% End of file merchant.m
```

```
%===============================================================================
%           Naval Postgraduate School - CA
%
%   Type   : Function
%   Name   : findInputMomSet.m
%   Function  : Finds the moment invariants input set for each view angle
%            This set will be used as the input for the neural network classifier
%   Date   01 march 2001
%   Version   : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%===============================================================================

function [trainingSet] = findInputMomSet( noise_silhouettes,rows,columns);

totalLines=12;   %number of moment functions
[X,totalColumns]=size(noise_silhouettes);
trainingSet= zeros(totalLines,totalColumns);
X1=zeros(rows,columns);

for i=1:totalColumns

     temp=noise_silhouettes(:,i);
     X1(:)=temp;
     trainingSet(1:6,i)=find_mom_functions(X1);    % solid silhouette
     X2=X1*255;                                       %[ 0 0 ...] max=255
     XX1=edge(X2,'prewitt');                          %only the edges [0 0
0 ...] max=1
     trainingSet(7:12,i)=find_mom_functions(XX1); %boundary
end
%===============================================================================
% End of file findInputMomSet.m
```

```
%==============================================================================
%          Naval Postgraduate School - CA
%
%   Type   : Function
%   Name   : find_mom_functions.m
%   Function : returns six functions values relating to the central moment
functions
%              invariant under rotation, translation,reflection and scale
%   Date   01 march 2001
%   Version  : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%==============================================================================

function NewMoments = find_mom_functions(pattern);

%FIND_MOM_FUNCTIONS returns six functions values relating to the central moment
functions
%invariant under rotation, translation,reflection and scale
%   [NewM2,NewM3,NewM4,NewM5,NewM6,NewM7] = find_mom_functions(pattern)

%find the second and third-order central moments
m_1_1=find_moment(1,1,pattern);
m_1_2=find_moment(1,2,pattern);
m_2_1=find_moment(2,1,pattern);
m_2_0=find_moment(2,0,pattern);
m_0_2=find_moment(0,2,pattern);
m_0_3=find_moment(0,3,pattern);
m_3_0=find_moment(3,0,pattern);

%find the moment functions invariant under rotation and reflection
M2=(m_2_0 - m_0_2)^2 + 4*(m_1_1^2);
M3=(m_3_0 - 3*m_1_2)^2 + (3*m_2_1 - m_0_3)^2;
M4=(m_3_0 + m_1_2)^2 + (m_2_1 +m_0_3)^2;
M5=(m_3_0  -  3*m_1_2)*(m_3_0  +  m_1_2)*((m_3_0  +  m_1_2)^2  -  3*(m_2_1  +
m_0_3)^2)+(3*m_2_1 - m_0_3)*(m_2_1 + m_0_3)*(3*((m_3_0 + m_1_2)^2) - (m_2_1 +
m_0_3)^2);
M6=(m_2_0 - m_0_2)*((m_3_0 + m_1_2)^2 - (m_2_1 + m_0_3)^2) + 4*m_1_1*(m_3_0 +
m_1_2)*(m_2_1 + m_0_3);
M7=(3*m_2_1 - m_0_3)*(m_3_0 + m_1_2)*((m_3_0 + m_1_2)^2 - 3*(m_2_1 + m_0_3)^2)
- (m_3_0 - 3*m_1_2)*(m_2_1 + m_0_3)*(3*(m_3_0 + m_1_2)^2 - (m_2_1 + m_0_3)^2);

%normalizing the moment functions under scale using the radius of gyration
r=sqrt(m_2_0 + m_0_2);

%the new moment function M2 through M7
NewM2 = M2/(r^4);
NewM3 = M3/(r^6);
NewM4 = M4/(r^7);
NewM5 = M5/(r^12);
NewM6 = M6/(r^8);
NewM7 = M7/(r^12);

NewMoments=[NewM2
   NewM3
   NewM4
   NewM5
   NewM6
   NewM7];
%==============================================================================
% End of file find_mom_functions.m
```

```
%===============================================================================
%          Naval Postgraduate School - CA
%
%  Type   : Function
%  Name   : find_moment.m
%  Function  : returns the central moment related to the indexes p and q
%  Date   01 march 2001
%  Version  : 1.0
%  Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%===============================================================================

function [m_p_q] = find_moment(p,q,pattern);

%FIND_MOMENT returns the central moment related to the indexes p and q.
%   [M_P_Q] = FIND_MOMENT(P,Q,PATTERN)

[NRows,NColumns]=size(pattern);
count=sum(sum(pattern,1),2);

%find the centroid values for this pattern

[um,vm]= find_centroid(pattern);

%index rows==>vi index columns==>ui
%  m_p_q=m_p_q+((ui-um)^p)*((vi-vm)^q)

[ui,vi]=meshgrid(0:(NColumns-1),0:(NRows-1));%ui=[0 1 2 ...    vi=[0 0 0 ...
                                             %    0 1 2 ...        1 1 1 ...
                                             %    0 1 2 ...        2 2 2 ...

m_p_q=((ui-um).^p).*((vi-vm).^q);
pattern=double(pattern);
m_p_q=m_p_q.*pattern;
m_p_q=sum(sum(m_p_q,1),2);
m_p_q=m_p_q/count;
%===============================================================================
% End of file finf_moment.m
```

```
%===========================================================================
%           Naval Postgraduate School - CA
%
%   Type   : Function
%   Name   : find_centroid.m
%   Function  : returns the centroid values for an image silhouette
%   Date   01 march 2001
%   Version  : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%===========================================================================

function [um,vm] = find_centroid(pattern);

%FIND_CENTROID returns the centroid values for an image frame PATTERN.
%   [VM,UM] = FIND_CENTROID(PATTERN)


%considering the solid silhouette or its boundary as a binary matrix(ones and
zeros)
pattern=double(pattern);
[NRows,NColumns]=size(pattern);

%the mean values um and vm are the centroid of the given pattern

%index rows==>vi index columns==>ui
%   m_p_q=m_p_q+((ui-um)^p)*((vi-vm)^q)

[ui,vi]=meshgrid(0:(NColumns-1),0:(NRows-1));%ui=[0 1 2 ...     vi=[0 0 0 ...
                                             %    0 1 2 ...         1 1 1 ...
                                             %    0 1 2 ...         2 2 2 ...

ui=ui.*pattern;
um=sum(sum(ui,1),2);
vi=vi.*pattern;
vm=sum(sum(vi,1),2);
count=sum(sum(pattern,1),2);
vm=vm/count;
um=um/count;
%===========================================================================
% End of file find_centroid.m
```

81

```
%===========================================================================
%          Naval Postgraduate School - CA
%   Type   : Main program
%   Name   : mainShipRecon
%   Function  : Create and trains a neural network responsible for recognizing
ship types
%               based on the moment invariants calculated for each viewangle
silhouette
%
%   Date   01 march 2001
%   Version   : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%===========================================================================

azAngleInc=15;      % ==>12 azimuth angles
eleAngleInc=15;%0,15,30,45 ==>four elev angles    % 90;%one elevation angle ==>
zero degree
totalAzimuth=180/azAngleInc;% ==>12
totalElevation=4;%=90/eleAngleInc;
totalColumns=totalAzimuth*totalElevation;%each column means one view
totalLines=12;  %number of moment functions

inputSet= zeros(totalLines,totalColumns);
rows=210;
columns=280;
silhouettes=zeros(rows*columns,totalColumns);

%    =========================
%    DEFINING THE MODEL PROBLEM
%    =========================

%    The script file FINDINPUTMOMSET defines a matrix inputSet
%     which contains the moment functions(12 values) for all views defined by
the
%     "Azimuth Angle Increment" and "Elevation Angle Increment" of the ship
model.

%    Each target vector has 5 elements with
%    all zeros, except for a single 1.  Aircraft Carrier has a 1 in the
%    first element, Destroyer in the second, Frigate in the third
%    , Point Sur in the fourth and Merchant in the fifth.


%aircraft carrier
[verts,faces]=aircarrier;
[inputSet,silhouettes]= findInputSet( verts,faces,azAngleInc,eleAngleInc);
[x,y]=size(inputSet);

temp=zeros(5,y);
temp(1,:)=1;%first line means aircraft carrier

targets=temp;
alphabet=inputSet;
all_silhouettes=silhouettes;

1
%destroyer
[verts,faces]=destroyer;
[inputSet,silhouettes]= findInputSet( verts,faces,azAngleInc,eleAngleInc);
[x,y]=size(inputSet);
```

82

```matlab
temp=zeros(5,y);
temp(2,:)=1;%second line means destroyer

targets=[targets,temp];
alphabet=[alphabet,inputSet];
all_silhouettes=[all_silhouettes,silhouettes];

2
%frigate
[verts,faces]=frigate;
[inputSet,silhouettes]= findInputSet( verts,faces,azAngleInc,eleAngleInc);
[x,y]=size(inputSet);
temp=zeros(5,y);
temp(3,:)=1;%third line means frigate

targets=[targets,temp];
alphabet=[alphabet,inputSet];
all_silhouettes=[all_silhouettes,silhouettes];

3
%point sur
[verts,faces]=pointsur;
[inputSet,silhouettes]= findInputSet( verts,faces,azAngleInc,eleAngleInc);
[x,y]=size(inputSet);
temp=zeros(5,y);
temp(4,:)=1;%fourth line means point sur

targets=[targets,temp];
alphabet=[alphabet,inputSet];
all_silhouettes=[all_silhouettes,silhouettes];

4
%merchant
[verts,faces]=merchant;
[inputSet,silhouettes]= findInputSet( verts,faces,azAngleInc,eleAngleInc);
[x,y]=size(inputSet);
temp=zeros(5,y);
temp(5,:)=1;%fifth line means point sur

targets=[targets,temp];
alphabet=[alphabet,inputSet];
all_silhouettes=[all_silhouettes,silhouettes];

save all_silhouettes all_silhouettes;
clear all_silhouettes;
save alphabet alphabet;
save targets targets;

clf;
figure(gcf)
echo on

[R,Q] = size(alphabet);
[S2,Q] = size(targets);

%pause % Strike any key to define the network...

%     ====================
%     DEFINING THE NETWORK
%     ====================
```

```
%    The ship recognition network will have 20 TANSIG
%    neurons in its hidden layer.

S1 = 20;
net = newff(minmax(alphabet),[S1 S2],{'logsig' 'logsig'},'traingdx');
net.LW{2,1} = net.LW{2,1}*0.01;
net.b{2} = net.b{2}*0.01;

%pause % Strike any key to train the network...

%    ====================
%    TRAINING THE NETWORK
%    ====================

net.performFcn = 'sse';        % Sum-Squared Error performance function
net.trainParam.goal = 0.1;     % Sum-squared error goal.
net.trainParam.show = 10000;%20;        % Frequency of progress displays (in
epochs).
net.trainParam.epochs = 600000;  % Maximum number of epochs to train.
net.trainParam.mc = 0.95;      % Momentum constant.

%    Training begins...please wait...

P = alphabet;
T = targets;

[net,tr] = train(net,P,T);
save net net
5
%    ...and finally finishes.
%============================================================================
% End of file mainShipRecon.m
```

```
%=============================================================================
%          Naval Postgraduate School - CA
%   Type   : Procedure
%   Name   : findInputSet
%   Function  : Creates the 3-D model, change the viewpoint of the 3-D model,
%               extract the silhouette for each training angle
%   Type   : Function
%   Date   20/may/2000
%   Version  : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%=============================================================================

function        [trainingSet,silhouettes]        =        findInputSet(
verts,faces,azAngleInc,eleAngleInc);

totalAzimuth=180/azAngleInc;
totalElevation=4;%90/eleAngleInc;
totalColumns=totalAzimuth*totalElevation;%each column means one view
totalLines=12;   %number of moment functions
trainingSet= zeros(totalLines,totalColumns);
rows=210;
columns=280;
silhouettes=zeros(rows*columns,totalColumns);
figNumber=figure( ...
      'Name','Silhouette', ...
      'Position',[120 120 280 210]);% matriz => 210rows X 280columns

figure(figNumber)

%Creates the 3-D model
patch('Vertices',verts,'Faces',faces);
view(3)
axis equal;
axis off
axis vis3d
lineCount=0;


for i=-90:azAngleInc:89       %azimuth popa until proa
   i
   %for j=0:eleAngleInc:89     %elevation
   for j=0:eleAngleInc:46      %elevation
      j
      lineCount=lineCount+1;

      %Changes the model viewpoint to the desired training angle
      view(i,j);
      [X,map]=capture;        %[65, 65, ...]
      X1=X-65;                %[0 0 ... ] max=1
      silhouettes(:,lineCount)=X1(:);
      trainingSet(1:6,lineCount)=find_mom_functions(X1);   % solid silhouette
      X2=X1*255;                                          %[ 0 0 ...] max=255
      XX1=edge(X2,'prewitt');                             %only the edges [0 0
0 ...] max=1
      trainingSet(7:12,lineCount)=find_mom_functions(XX1); %boundary
   end
end
%=============================================================================
% End of file findInputSet.m
```

```
%=============================================================================
%          Naval Postgraduate School - CA
%   Type   : Function
%   Name   : interface.m
%   Function  : Create a Graphical User Interface in order to evaluate the
%               Automatic Target Recognition System implemented
%               3 ship silhouettes are shown in the interface:
%                      - the original silhoette
%                         - the rotated, scaled and noisy silhouette defined by the
user
%                      - the neural network "guessed" silhouette
%   Date   01 march 2001
%   Version   : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%=============================================================================

function interface(action)

if nargin<1,
    action='initialize';
end;

if strcmp(action,'initialize'),
    oldFigNumber=watchon;

    figNumber=figure( ...
        'Name','Neural Network Ship Silhouette Recognition', ...
        'NumberTitle','off', ...
        'Visible','off', ...
        'BackingStore','off');

    axPos=[0.40 0.95-0.28 0.20 0.28];
    axHndl1=axes( ...
        'Units','normalized', ...
        'Position',axPos, ...
        'XTick',[],'YTick',[], ...
        'Box','on');
    labelPos=[0.05 0.80 0.30 0.05];
    uicontrol( ...
        'Style','text', ...
        'String','Original Ship Silhouette', ...
        'BackgroundColor','k', ...
        'ForegroundColor','w', ...
        'Units','normalized', ...
        'Position',labelPos);

    axHndl2=axes( ...
        'Units','normalized', ...
        'Position',axPos+[0 -0.31 0 0], ...
        'XTick',[],'YTick',[], ...
        'Box','on');
    labelPos=[0.05 0.50 0.30 0.05];
    uicontrol( ...
        'Style','text', ...
        'String','Ship Silhouette with Noise', ...
        'BackgroundColor','k', ...
        'ForegroundColor','w', ...
        'Units','normalized', ...
        'Position',labelPos);
```

```matlab
axHndl3=axes( ...
    'Units','normalized', ...
    'Position',axPos+[0 -0.62 0 0], ...
    'XTick',[],'YTick',[], ...
    'Box','on');
labelPos=[0.05 0.20 0.30 0.05];
uicontrol( ...
    'Style','text', ...
    'String','Network''s Guess', ...
    'BackgroundColor','k', ...
    'ForegroundColor','w', ...
    'Units','normalized', ...
    'Position',labelPos);

%==================================
% Information for all buttons
top=0.95;
bottom=0.05;
labelColor=[0.8 0.8 0.8];
btnWid=0.20;
btnHt=0.10;
right=0.95;
left=right-btnWid;
% Spacing between the button and the next command's label
spacing=0.05;

%==================================
% The CONSOLE frame
frmBorder=0.02;
frmPos=[left-frmBorder         bottom-frmBorder         btnWid+2*frmBorder
0.9+2*frmBorder];
h=uicontrol( ...
    'Style','frame', ...
    'Units','normalized', ...
    'Position',frmPos, ...
    'BackgroundColor',[0.5 0.5 0.5]);

%==================================
% The NEW SILHOUETTE button
btnNumber=1;
yPos=top-btnHt-(btnNumber-1)*(btnHt+spacing);
labelStr='New Silhouette';
callbackStr='interface(''new'');';

% Generic button information
btnPos=[left yPos btnWid btnHt];
flyHndl=uicontrol( ...
    'Style','pushbutton', ...
    'Units','normalized', ...
    'Position',btnPos, ...
    'String',labelStr, ...
    'Callback',callbackStr);

%==================================
% The NOISE slider
btnNumber=2;
yPos=top-btnHt-(btnNumber-1)*(btnHt+spacing);
labelStr='Noise';

% Generic button information
```

```matlab
sldPos=[left yPos btnWid btnHt/2];
labelPos=[left yPos+btnHt/2 btnWid btnHt/2];
sldHndl=uicontrol( ...
    'Style','slider', ...
    'Units','normalized', ...
    'Position',sldPos);

uicontrol( ...
    'Style','text', ...
    'Units','normalized', ...
    'String','Noise', ...
    'Position',labelPos);

%=================================
% The ROTATION slider
btnNumber=3;
yPos=top-btnHt-(btnNumber-1)*(btnHt+spacing);
labelStr='Rotation';

% Generic button information
rotPos=[left yPos btnWid btnHt/2];
labelPos=[left yPos+btnHt/2 btnWid btnHt/2];
rotHndl=uicontrol( ...
    'Style','slider', ...
    'Units','normalized', ...
    'Position',rotPos);

uicontrol( ...
    'Style','text', ...
    'Units','normalized', ...
    'String','Rotation', ...
    'Position',labelPos);

%=================================
% The SCALE slider
btnNumber=4;
yPos=top-btnHt-(btnNumber-1)*(btnHt+spacing);
labelStr='Scale';

% Generic button information
sclPos=[left yPos btnWid btnHt/2];
labelPos=[left yPos+btnHt/2 btnWid btnHt/2];
sclHndl=uicontrol( ...
    'Style','slider', ...
    'Units','normalized', ...
    'Position',sclPos);

uicontrol( ...
    'Style','text', ...
    'Units','normalized', ...
    'String','Scale', ...
    'Position',labelPos);


%=================================
% The INFO button
labelStr='Info';
callbackStr='interface(''info'')';
infoHndl=uicontrol( ...
    'Style','push', ...
```

```matlab
        'Units','normalized', ...
        'Position',[left bottom+btnHt+spacing btnWid btnHt], ...
        'String',labelStr, ...
        'Callback',callbackStr);

    %===================================
    % The CLOSE button
    labelStr='Close';
    callbackStr='close(gcf)';
    closeHndl=uicontrol( ...
        'Style','push', ...
        'Units','normalized', ...
        'Position',[left bottom btnWid btnHt], ...
        'String',labelStr, ...
        'Callback',callbackStr);

    % Uncover the figure
    hndlList=[axHndl1 axHndl2 axHndl3 sldHndl rotHndl sclHndl];
    set(figNumber, ...
    'Visible','on', ...
    'UserData',hndlList);

    watchoff(oldFigNumber);
    choice=0;
    interface new;
    figure(figNumber);


elseif strcmp(action,'new'),
    figNumber=watchon;
    hndlList=get(figNumber,'Userdata');
    axHndl1=hndlList(1);
    axHndl2=hndlList(2);
    axHndl3=hndlList(3);
    sldHndl=hndlList(4);
    rotHndl=hndlList(5);
    sclHndl=hndlList(6);

    load all_silhouettes;
    load rows;
    load columns;
    load net;

    [X,Y]=size(all_silhouettes);
    load choice;
    choice=choice+1;
    save choice choice;
    choice
    %find a random silhouette as the original one
    randsilhouette=all_silhouettes(:,choice);

    %find the corresponding silhouette after rotation
    rotsilhouette=randsilhouette;
    rotationlevel=get(rotHndl,'Value');
    rotationlevel*180
    X1=zeros(rows,columns);
    X1(:)=randsilhouette;
    X2=imrotate(X1,(rotationlevel*180),'bilinear','crop');
    rotsilhouette=X2(:);
```

```matlab
    %find the corresponding silhouette after scaling
    sclsilhouette=rotsilhouette;
    scalelevel=get(sclHndl,'Value')
    X1=zeros(rows,columns);
    X1(:)=rotsilhouette;
    X2=imrotate(X1,45,'bilinear','crop');
    sclsilhouette=X2(:);

    %find the corresponding silhouette after adding noise
    noiselevel=get(sldHndl,'Value')
    noise=round(randn(rows*columns,1)*noiselevel);
    y=noise>0;
    w=noise<0;
    noise=y+w;
    noisesilhouette=rotsilhouette+noise;

    testsilhouette=noisesilhouette;
    testMoments= findInputMomSet( testsilhouette,rows,columns);
    A = sim(net,testMoments);
    output=compet(A);
    result=find(output==1)-1;
    result
    outsilhouette=all_silhouettes(:,48*result+25);

    axes(axHndl1);
    plotSilhouette(randsilhouette,rows,columns);
    axes(axHndl2);
    plotSilhouette(testsilhouette,rows,columns);
    axes(axHndl3);
    plotSilhouette(outsilhouette,rows,columns);
    watchoff(figNumber);

elseif strcmp(action,'info'),
    ttlStr=get(gcf,'Name');
    hlpStr=  {'                                                    '
        ' This window demonstrates the use of a neural      '
        ' network to recognize the ship silhouettes.        '
        ' of the alphabet. The system used here is based    '
        ' on a two layer network (not including the input   '
        ' layer) with 20 neurons in the hidden layer and    '
        ' 5 neurons (one for each ship type) in the output  '
        ' layer. The moment invariants values are the input '
        ' of the neural network. They are 12 element vectors'
        ' representing the invariants for the silhouette.    '
        '                                                    '
        ' The network has already been trained using        '
        ' backpropagation - you can test it by pressing     '
        ' the "New Letter" button. This passes a random     '
        ' letter to the network. The "Noise" slider adds    '
        ' random noise to make the classification problem   '
        ' more difficult. The "Rotation" and the "Scale"    '
        ' buttons allow you to rotate and to change the     '
        ' scale of the silhouette to be tested.             '
        '                                                    '
        ' File name: interface.m                            '};
    helpwin(hlpStr,ttlStr);

end;
%==============================================================================
% End of file interface.m
```

```
%================================================================
%           Naval Postgraduate School - CA
%   Type   : Function
%   Name   : plotSilhouette.m
%   Function : Draws   Ship Silhouette inside the Graphical User Interface
%   Date   01 march 2001
%   Version   : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%================================================================

function plotSilhouette(silhouette,rows,columns)

X1=zeros(rows,columns);
X1(:)=silhouette;
imagesc(X1);%body silhouette
colormap(gray(2))
set(gca,'XTick',[],'YTick',[]);
%================================================================
% End of file plotSilhouette.m
```

```
%=============================================================================
%          Naval Postgraduate School - CA
%   Type   : Function
%   Name   : segmentation.m
%   Function : Segments a FLIR real image using a histogram and threshold
technique
%   Date   01 march 2001
%   Version   : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%=============================================================================

%Aircraft Carrier
[X,Map]=imread('carrier10.tif','tif');% 240 lines by 320 columns
%[X,Map]=imread('carrier11.tif','tif');% 240 lines by 320 columns
%[X,Map]=imread('carrier3.tif','tif');% 240 lines by 320 columns
%[X,Map]=imread('carrier6.tif','tif');% 240 lines by 320 columns

%Destroyer
%[X,Map]=imread('ab6.tif','tif');% 240 lines by 320 columns
%[X,Map]=imread('dd1.tif','tif');% 240 lines by 320 columns

%Merchant
%[X,Map]=imread('group3tanker.tif','tif');% 240 lines by 320 columns
%[X,Map]=imread('group3tanker2.tif','tif');% 240 lines by 320 columns
%[X,Map]=imread('flir2_2.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir2_3.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir2_4.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir2_5.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir2_6.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir7_1.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir7_11.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir7_12.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir8_1.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir8_12.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir8_15.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir8_2.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('flir7_5.jpg','jpg');% 240 lines by 320 columns

%Research Ship
%[X,Map]=imread('iranpb1.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('iranpb2.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('iranpb3.jpg','jpg');% 240 lines by 320 columns
%[X,Map]=imread('pointsur.bmp','bmp');% 240 lines by 320 columns


X1=X(:,:,1);
figure
colormap(gray(256))
imshow(X1)

[counts,X]=imhist(X1);
figure
stem(X,counts)
p=polyfit(X,counts,30);
r=roots(p);
maximo=max(counts);
threshold=find(counts==maximo)
rr=(r<threshold).*r;
newThreshold=abs(max(real(rr)))+10
[x,y]=size(X1);
```

```matlab
XX1=double(X1);
Y=(XX1<newThreshold).*(XX1>10);
Y=double(Y);
Y=Y*255;

figure
colormap(gray(256))
imshow(Y)

resImg2 = bwmorph(Y,'spur');
figure
imshow(resImg2);
colormap(gray(256));

resImg3 = bwmorph(resImg2,'clean');
figure
imshow(resImg3);
colormap(gray(256));

resImg4 = bwmorph(resImg3,'fill');
figure
imshow(resImg4);
colormap(gray(256));

[resImg5,maps1]=bwlabel(resImg4,4);
resImg6=zeros(size(Y));
maximo=0;
for i=1:maps1
    tempMap=(resImg5==i);
    total=sum(sum(tempMap));
    if (total>maximo)
        resImg6=tempMap;
        maximo=total;
    end
end
figure
colormap(gray(256))
imshow(resImg6)

resImg7 = bwfill(resImg6,'holes');
figure
imshow(resImg7);
colormap(gray(256));

%Original is 240x320 but I need to save 210x280
X1=resImg7(16:(x-15),41:y);
[x,y]=size(X1);

ab1=zeros(x*y,1);
ab1=resImg7(:);

save ab1 ab1;
%==========================================================================
% End of file segmentation.m
```

```
%=============================================================================
%           Naval Postgraduate School - CA
%
%   Type   : Main program
%   Name   : createTestSet.m
%   Function : Creates 05 mat files containing the silhouettes
%                    of each ship for increments of one degree in azimuth and
elevation
%              then plots the errors
%   Date   01 march 2001
%   Version  : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%=============================================================================

azAngleInc=1;%5; % ==> 36 azimuth angles
eleAngleInc=15;% ==> 10 elevation angles
totalAzimuth=180/azAngleInc;% ==>180
totalElevation=2%4;%10;
totalColumns=totalAzimuth*totalElevation;%each  column  means  one  view  ==>
36*10=360 silhouettes
totalLines=12;  %number of moment functions

rows=210;
columns=280;

resultSet=zeros(totalAzimuth,totalElevation);

1
%aircraft carrier
[verts,faces]=aircarrier;
rightResult=1;
resultSet= findResultSet( verts,faces,azAngleInc,eleAngleInc,rightResult);

aircarrierResultSet=resultSet;
save aircarrierResultSet aircarrierResultSet;
clear aircarrierResultSet;

2
%destroyer
[verts,faces]=destroyer;
rightResult=2;
resultSet= findResultSet( verts,faces,azAngleInc,eleAngleInc,rightResult);

destroyerResultSet=resultSet;
save destroyerResultSet destroyerResultSet;
clear destroyerResultSet;

3
%frigate
[verts,faces]=frigate;
rightResult=3;
resultSet= findResultSet( verts,faces,azAngleInc,eleAngleInc,rightResult);

frigateResultSet=resultSet;
save frigateResultSet frigateResultSet;
clear frigateResultSet;

4
%point sur
[verts,faces]=pointsur;
```

```
rightResult=4;
resultSet= findResultSet( verts,faces,azAngleInc,eleAngleInc,rightResult);

pointsurResultSet=resultSet;
save pointsurResultSet pointsurResultSet;
clear pointsurResultSet;

5
%merchant
[verts,faces]=merchant;
rightResult=5;
resultSet= findResultSet( verts,faces,azAngleInc,eleAngleInc,rightResult);

merchantResultSet=resultSet;
save merchantResultSet merchantResultSet;
clear merchantResultSet;

pause

load aircarrierResultSet1
temp=aircarrierResultSet;
load aircarrierResultSet1
temp=[temp,aircarrierResultSet];
aircarrierResultSet=temp;
save aircarrierResultSet aircarrierResultSet;

load destroyerResultSet1
temp=destroyerResultSet;
load destroyerResultSet1
temp=[temp,destroyerResultSet];
destroyerResultSet=temp;
save destroyerResultSet destroyerResultSet;

load frigateResultSet1
temp=frigateResultSet;
load frigateResultSet1
temp=[temp,frigateResultSet];
frigateResultSet=temp;
save frigateResultSet frigateResultSet;

load pointsurResultSet1
temp=pointsurResultSet;
load pointsurResultSet1
temp=[temp,pointsurResultSet];
pointsurResultSet=temp;
save pointsurResultSet pointsurResultSet;

load merchantResultSet1
temp=merchantResultSet;
load merchantResultSet1
temp=[temp,merchantResultSet];
merchantResultSet=temp;
save merchantResultSet merchantResultSet;

%aircraft carrier
figure
azimuthAngle=[-90:1:89];
errorAircarrier=sum(aircarrierResultSet');
errorPercent=(errorAircarrier/4)*100;
```

```
newAzimuthAngle=[-90,-85,-80,-75,-70,-65,-60,-55,-50,-45,-30,-
15,0,15,30,45,50,55,60,65,70,75,80,85];
newErrorPercent1=errorPercent(1:5:44);
newErrorPercent2=errorPercent(45:15:136);
newErrorPercent3=errorPercent(140:5:179);
newErrorPercent=[newErrorPercent1,newErrorPercent2,newErrorPercent3];

plot(newAzimuthAngle,newErrorPercent,'or')
hold
plot(azimuthAngle,errorPercent)
xlabel('azimuth angle')
ylabel('error(%)')
title('Aircraft Carrier Testing Results (adding all 4 elevation angles)')
legend('Training  set  ==>  96','Testing  set  ==>  720','Silhouettes  correct
classified ==> 642 (89.2%)');

figure
elevationAngle=[0:15:46];
errorAircarrier=sum(aircarrierResultSet);
errorPercent=(errorAircarrier/180)*100;
newElevationAngle=[0:15:46];
newErrorPercent=errorPercent(1:1:4);
plot(newElevationAngle,newErrorPercent,'or')
hold
plot(elevationAngle,errorPercent)
xlabel('elevation angle')
ylabel('error(%)')
title('Aircraft Carrier Testing Results (adding all 180 azimuth angles)')
legend('Training  set  ==>  96','Testing  set  ==>  720','Silhouettes  correct
classified ==> 642');

%Destroyer
figure
azimuthAngle=[-90:1:89];
errorDestroyer=sum(destroyerResultSet');
errorPercent=(errorDestroyer/4)*100;

newAzimuthAngle=[-90,-85,-80,-75,-70,-65,-60,-55,-50,-45,-30,-
15,0,15,30,45,50,55,60,65,70,75,80,85];
newErrorPercent1=errorPercent(1:5:44);
newErrorPercent2=errorPercent(45:15:136);
newErrorPercent3=errorPercent(140:5:179);
newErrorPercent=[newErrorPercent1,newErrorPercent2,newErrorPercent3];

plot(newAzimuthAngle,newErrorPercent,'or')
hold
plot(azimuthAngle,errorPercent)
xlabel('azimuth angle')
ylabel('error(%)')
title('Destroyer Testing Results (adding all 4 elevation angles)')
legend('Training  set  ==>  96','Testing  set  ==>  720','Silhouettes  correct
classified ==> 698 (96.9%)');

figure
elevationAngle=[0:15:46];
errorDestroyer=sum(destroyerResultSet);
errorPercent=(errorDestroyer/180)*100;
newElevationAngle=[0:15:46];
newErrorPercent=errorPercent(1:1:4);
plot(newElevationAngle,newErrorPercent,'or')
```

```
hold
plot(elevationAngle,errorPercent)
xlabel('elevation angle')
ylabel('error(%)')
title('Destroyer Testing Results (adding all 180 azimuth angles)')
legend('Training   set  ==>   96','Testing   set  ==>   720','Silhouettes   correct
classified ==> 698');


%Frigate
figure
azimuthAngle=[-90:1:89];
errorFrigate=sum(frigateResultSet');
errorPercent=(errorFrigate/4)*100;
newAzimuthAngle=[-90,-85,-80,-75,-70,-65,-60,-55,-50,-45,-30,-
15,0,15,30,45,50,55,60,65,70,75,80,85];
newErrorPercent1=errorPercent(1:5:44);
newErrorPercent2=errorPercent(45:15:136);
newErrorPercent3=errorPercent(140:5:179);
newErrorPercent=[newErrorPercent1,newErrorPercent2,newErrorPercent3];
plot(newAzimuthAngle,newErrorPercent,'or')
hold
plot(azimuthAngle,errorPercent)
xlabel('azimuth angle')
ylabel('error(%)')
title('Frigate Testing Results (adding all 4 elevation angles)')
legend('Training   set  ==>   96','Testing   set  ==>   720','Silhouettes   correct
classified ==> 666 (92,5%)');

figure
elevationAngle=[0:15:46];
errorFrigate=sum(frigateResultSet);
errorPercent=(errorFrigate/180)*100;
newElevationAngle=[0:15:46];
newErrorPercent=errorPercent(1:1:4);
plot(newElevationAngle,newErrorPercent,'or')
hold
plot(elevationAngle,errorPercent)
xlabel('elevation angle')
ylabel('error(%)')
title('Frigate Testing Results (adding all 180 azimuth angles)')
legend('Training   set  ==>   96','Testing   set  ==>   720','Silhouettes   correct
classified ==> 666');



%Pointsur
figure
azimuthAngle=[-90:1:89];
errorPointsur=sum(pointsurResultSet');
errorPercent=(errorPointsur/4)*100;
newAzimuthAngle=[-90,-85,-80,-75,-70,-65,-60,-55,-50,-45,-30,-
15,0,15,30,45,50,55,60,65,70,75,80,85];
newErrorPercent1=errorPercent(1:5:44);
newErrorPercent2=errorPercent(45:15:136);
newErrorPercent3=errorPercent(140:5:179);
newErrorPercent=[newErrorPercent1,newErrorPercent2,newErrorPercent3];
plot(newAzimuthAngle,newErrorPercent,'or')
hold
plot(azimuthAngle,errorPercent)
```

```matlab
xlabel('azimuth angle')
ylabel('error(%)')
title('Pointsur Testing Results (adding all 4 elevation angles)')
legend('Training  set  ==>  96','Testing  set  ==>  720','Silhouettes  correct
classified ==> 644 (89.4%)');

figure
elevationAngle=[0:15:46];
errorPointsur=sum(pointsurResultSet);
errorPercent=(errorPointsur/180)*100;
newElevationAngle=[0:15:46];
newErrorPercent=errorPercent(1:1:4);
plot(newElevationAngle,newErrorPercent,'or')
hold
plot(elevationAngle,errorPercent)
xlabel('elevation angle')
ylabel('error(%)')
title('Pointsur Testing Results (adding all 180 azimuth angles)')
legend('Training  set  ==>  96','Testing  set  ==>  720','Silhouettes  correct
classified ==> 644');

%Merchant
figure
azimuthAngle=[-90:1:89];
errorMerchant=sum(merchantResultSet');
errorPercent=(errorMerchant/4)*100;
newAzimuthAngle=[-90,-85,-80,-75,-70,-65,-60,-55,-50,-45,-30,-
15,0,15,30,45,50,55,60,65,70,75,80,85];
newErrorPercent1=errorPercent(1:5:44);
newErrorPercent2=errorPercent(45:15:136);
newErrorPercent3=errorPercent(140:5:179);
newErrorPercent=[newErrorPercent1,newErrorPercent2,newErrorPercent3];
plot(newAzimuthAngle,newErrorPercent,'or')
hold
plot(azimuthAngle,errorPercent)
xlabel('azimuth angle')
ylabel('error(%)')
title('Merchant Testing Results (adding all 4 elevation angles)')
legend('Training  set  ==>  96','Testing  set  ==>  720','Silhouettes  correct
classified ==> 634 (88.1%)');

figure
elevationAngle=[0:15:46];
errorMerchant=sum(merchantResultSet);
errorPercent=(errorMerchant/180)*100;
newElevationAngle=[0:15:46];
newErrorPercent=errorPercent(1:1:4);
plot(newElevationAngle,newErrorPercent,'or')
hold
plot(elevationAngle,errorPercent)
xlabel('elevation angle')
ylabel('error(%)')
title('Merchant Testing Results (adding all 180 azimuth angles)')
legend('Training  set  ==>  96','Testing  set  ==>  720','Silhouettes  correct
classified ==> 634');
%=========================================================================
% End of file createTestSet.m
```

```
%=========================================================================
%          Naval Postgraduate School - CA
%   Type  : Procedure
%   Name  : findResultSet.m
%   Function : returns a vector with the size of all the viewangles being
tested
%                 where "1" will mean misclassified and "0" will mean correct
classified
%   Date   01 march 2001
%   Version   : 1.0
%   Author : Jorge Amaral Alves, LCDR (Brazilian Navy)
%=========================================================================

function              resultSet              =              findResultSet(
verts,faces,azAngleInc,eleAngleInc,rightResult);

totalAzimuth=180/azAngleInc;
totalElevation=2;%4;%10;%90/eleAngleInc;
totalColumns=totalAzimuth*totalElevation;%each column means one view
totalLines=12;   %number of moment functions

resultSet= zeros(totalAzimuth,totalElevation);

trainingSet= zeros(totalLines,1);

rows=210;
columns=280;
silhouettes=zeros(rows*columns,totalColumns);


figNumber=figure( ...
      'Name','Silhouette', ...
      'Position',[120 120 280 210]);% matriz => 210rows X 280columns

figure(figNumber)
patch('Vertices',verts,'Faces',faces);
view(3)
axis equal;
axis off
axis vis3d
lineCount=1;

load net;

azimuthCount=0;
for i=-90:azAngleInc:89       %azimuth popa until proa
   i
   azimuthCount=azimuthCount+1;
   elevationCount=0;
   %for j=0:eleAngleInc:16%46      %elevation
   for j=30:eleAngleInc:46      %elevation
      j
      elevationCount=elevationCount+1;
      view(i,j);
      [X,map]=capture;          %[65, 65, ...
      X1=X-65;                  %[0 0 ... ] max=1

      trainingSet(1:6,lineCount)=find_mom_functions(X1);   % solid silhouette
      X2=X1*255;                                            %[ 0 0 ...] max=255
```

99

```matlab
      XX1=edge(X2,'prewitt');                              %only the edges [0 0
0 ...] max=1
      trainingSet(7:12,lineCount)=find_mom_functions(XX1); %boundary

      A = sim(net,trainingSet);
      output=compet(A);
      result=find(output==1);%-1;
      result
      if result==rightResult
          resultSet(azimuthCount,elevationCount)=0;
      else
          resultSet(azimuthCount,elevationCount)=1;
      end
    end
end
%=========================================================================
% End of file findResultSet.m
```

LIST OF REFERENCES

1. Richard, C., Hemani, H., "Identification of three-dimensional objects using Fourier descriptors of the boundary curve", *IEEE-T Systems Man. Cybernet.*, SMC-4, Vol. 4, pp. 371-378, 1974.

2. Bebis, G. N., Papadourakis, G. M., "Object Recognition using invariant object boundary representations and neural network models", *Pattern Recognition 25*, Vol. 1, pp. 25-44, 1992.

3. Ettinger, G., "Hierarchical object recognition using libraries of parameterized model sub-parts", Master's Thesis, MIT, 1987.

4. Dubois, S., Glanz, F., "An autoregressive model approach to two-dimensional shape classification", *IEEE-T on PAMI*, Vol. 8, pp. 55-66, 1986.

5. Zahn, C. T., Roskies, R. Z., "Fourier descriptors for plane closed curves", *IEEE-T Comput.21*, Vol. 3, pp. 269-281, 1972.

6. Gorman, J., Mitchell, R., Kuhl, F., "Partial shape recognition using dynamic programming", *IEEE-T on PAMI 10*, Vol. 2, pp. 257-266, 1988.

7. Sadjadi, F., "Automatic object recognition: critical issues and current approaches", *Proc. SPIE 1471*, pp. 303-313, 1991.

8. Jaggi, S., Karl, C., Mallat, S., Willsky, A., "Silhouette recognition using high-resolution pursuit", *Pattern Recognition 32*, pp. 753-771, 1999.

9. Hu, M. K., "Visual pattern recognition by moment invariants", *IRE Trans. On Information Theory*, Vol. 8, pp. 179-187, 1962.

10. Casasend, D., Pauly, J., Fetterly, D., "IR ships classification using a new moment pattern recognition concept", *Infrared Technology for Target Detection and Classification*, SPIE Vol. 302, pp. 126-133, 1981.

11. Rogers, S. K., Ruck, D. W., Kabrisky, M., Tarr, G. L., "Artificial neural networks for automatic target recognition", *Applications of Artificial Neural Networks*, SPIE Vol. 1294, pp. 1-12, 1990.

12. Teh, C., Chin, R., "On image analysis by the method of moments", *IEEE-T on PAMI*, Vol. 10, pp. 291-310, 1988.

13. Kashyap, R., Chellapa, R., "Stochastic models for closed boundary analysis: representation and reconstruction", *IEEE-T Inf. Theory*, Vol. 27, pp. 109-119, 1981.

14. Freeman, H., "Shape description via the use of critical points", *Pattern Recognition*, Vol. 10, pp. 159-166, 1978.

15. Jaggi, S., "Multiscale geometric feature extraction and object recognition", Ph.D.Thesis, Massachusetts Institute of Technology, 1997.

16. Dudani, S. A., Breeding, K. J., McGhee, R.B., "Aircraft identification by moment invariants", *IEEE-T on Computers*, Vol. 26, No.1, pp. 39-46, Jan. 1977.

17. Wallace, T. P., Wintz, P., "An efficient, three-dimensional aircraft recognition algorithm using normalized Fourier descriptors", *Comput.Graphics Image Proc.*, Vol. 3, pp. 99-126, 1980.

18. Reeves, A. P., Prokop, R. J., Andrews, S. E., Kuhl, F. P., "Three-dimensional shape analysis using moments and Fourier descriptors", *IEEE-T on PAMI*, Vol. 10, No. 6, pp. 937-943, 1988.

19. Khotanzad, A., Lu, J., "Classification of invariant image representations using a neural network", *IEEE-T on Acoustics, Speech, and Signal Processing*, Vol. 38, No. 6, pp. 1028-1038, 1990.

20. Reddi, S. S., "Radial and angular moment invariants for image identification", *IEEE-T on PAMI*, Vol. 3, pp. 240-242, 1981.

21. Lippman, R., "An introduction to computing with neural nets", *IEEE ASSP Mag.*, pp. 109-119, April 1987.

22. Carpenter, G., "Neural network models for pattern recognition and associative memory", *Neural Networks*, Vol. 2, pp. 243-257, 1989.

23. Fahlman, S., Hilton, G., "Connectionist architectures for artificial intelligence", *IEEE Comput.*, pp. 100-109, 1987.

24. Sejnowski, T., Rosenberg, C., *NETtalk: a parallel network that learns to read aloud*, J.A.Anderson and E.Rosenfeld Neurocomputing Foundations, MIT Press, Cambridge, MA, 1988.

25. Perantonis, S. J., Lisboa, J. G., "Translation, rotation, and scale invariant pattern recognition by high-order neural network and moment classifiers", *IEEE-T on Neural Networks*, Vol. 3, No. 2, pp. 241-251, March 1992.

26. Papadourakis, G. M., Bebis, G., Georgiopoulos, M., "Machine printed character recognition using neural networks", *Int. Neural Network Conf.*, Paris, 1990.

27. Touretzky, D., Pomerleau, D., "What's hidden in the hidden layers?", *Byte Mag.*, pp. 227-233, 1989.

28. Rumelhart, D. E., McClelland, J. L. and the PDP, *Explorations in the Microstructure of cognition*, Vol. 1: Foundations, MIT Press, Cambridge, MA, 1986.

29. Bebis, G., Papadourakis, G. M., Georgiopoulos, M., "Backpropagation: increasing rate of convergence by predictable pattern loading", *Intell. Syst*, Rev. 1, pp. 14-30, 1989.

30. Jane's Information Group Ltd, *Jane's Fighting Ships*, 1999.

31. Jane's Information Group Ltd, *Jane's Merchant Ships*, 1999.

THIS PAGE INTENTIONALLY LEFT BLANK

BIBLIOGRAPHY

Anderson, James A., *An Introduction to Neural Networks*, Third Printing, A Bradford Book, MIT Press, 1997.

Russ, John C., *The Image Processing Handbook*, Third Edition, CRC Press with IEEE Press, 1998.

The MathWorks Inc., *Learning MATLAB V.5.3*, 1999.

Looney, Carl G., *Pattern Recognition Using Neural Networks Theory and Algorithms for Engineers and Scientists*, Oxford University Press, 1997.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
    8725 John J. Kingman Rd., STE 0944
    Ft. Belvoir, Virginia 22060-6218

2. Dudley Knox Library 2
    Naval Postgraduate School
    411 Dyer Rd.
    Monterey, California 93943-5101

3. National Reconnaissance Office 1
    14675 Lee Road
    Chantilly, Virginia 20151-1715

4. Professor Neil C. Rowe, Code CS/Nr 2
    Department of Computer Science
    Naval Postgraduate School
    Monterey, California 93943-5000

5. Prof. Robert B. McGhee, Code CS/Mz 1
    Department of Electrical and Computer Engineering
    Naval Postgraduate School
    Monterey, California 93943-5121

6. Director, Instituto de Pesquisas da Marinha 1
    Rua Ipiru 2, Ilha do Governador
    21931-090 Rio de Janeiro - RJ
    BRAZIL

7. Head of Research, Instituto de Pesquisas da Marinha 1
    Rua Ipiru 2, Ilha do Governador
    21931-090 Rio de Janeiro - RJ
    BRAZIL

8. Diretoria de Ensino da Marinha 1
    via: Brazilian Naval Commission
    5130 MacArthur Boulevard, NW
    Washington, D.C. 20016-3344

9. Diretoria de Telecomunicações da Marinha 1
    via: Brazilian Naval Commission
    5130 MacArthur Boulevard, NW
    Washington, D.C. 20016-3344

10. Director, Diretoria de Engenharia Naval 1
Rua Primeiro de Março, 118 - 10º andar - Centro
20.010-000 Rio de Janeiro - RJ
BRAZIL

11. Diretoria de Engenharia Naval, Library 1
Rua Primeiro de Março, 118 - 10º andar – Centro
20.010-000 Rio de Janeiro - RJ
BRAZIL

12. Diretoria de Sistemas de Armas da Marinha, Library 1
Rua Primeiro de Março, 118 - 19º andar - Centro
20.010-000 Rio de Janeiro - RJ
BRAZIL

13. Instituto Militar de Engenharia, Library 1
Praça General Tibúrcio 80, Praia Vermelha
22290-270 Rio de Janeiro - RJ
BRAZIL

14. Centro Técnico Aeroespacial, Library 1
Praça Mal. Eduardo Gomes 50, Vila das Acácias
12228-904 São José dos Campos - SP
BRAZIL

15. CC(EN) Jorge Amaral Alves 3
Rua Acacio Santos 110, Osvaldo Cruz
21550-250 Rio de Janeiro - RJ
BRAZIL